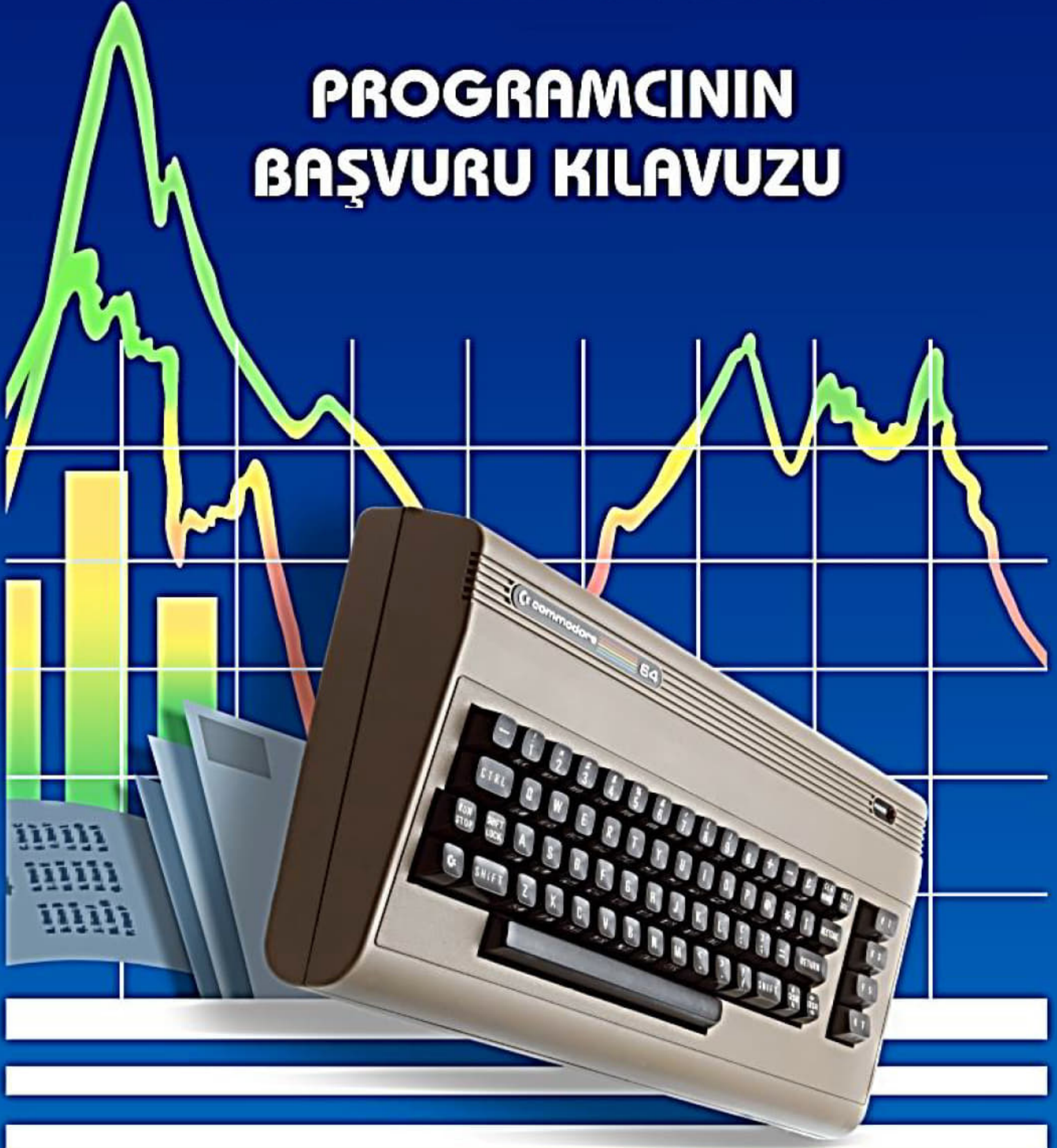
COMMODORE 64

PROGRAMCININ BAŞVURU KILAVUZU



Commodore
Bilgisayar



PROGRAMCININ

BAŞVURU KILAVUZU

İLK BASKI	1982
SON BASKI	2025
DÜZENLEME	Recep MUM
REVİZYON	R.22.11.2025.0008

İÇİNDEKİLER

BAŞLARKEN IX
Kapsamı nedir? X
Bu rehber nasıl kullanılır? XI
Commodore 64 uygulama kılavuzu XII
Commodore 64 bilgi ağı XVI

1. BASIC PROGRAMLAMA KURALLARI 1
Giriş 2
Ekran gösterim kodları (BASIC temel karakter seti) 2
İşletim sistemi (OS) 2
Sabit ve değişken değerlerin programlanması 5
Tam sayı, ondalık sayı ve yazınsal dizi sabitleri 5
Tam sayı, ondalık sayı ve yazınsal dizi değişkenleri 8
Tam sayı, ondalık sayı ve yazınsal dizi dizeleri 9
İfadeler (expressions) ve operatörler 12
Aritmetik ifadeler 13
Aritmetik işlemler 13
Bağıntı (relational) operatörler 14
Mantıksal operatörler 15
İşlemlerin öncelik sırası 17
Yazınsal dizi (string) işlemleri 20
Yazınsal dizi ifadeleri 20
Programlama teknikleri 21
Veri dönüşümleri 21
INPUT yönergesinin kullanımı 21
GET yönergesinin kullanımı 25
BASIC programlarını nasıl kısaltabiliriz? 27

2. BASIC DİLİ SÖZLÜĞÜ 31
Giriş 32
BASIC anahtar-sözcükleri, kısaltmalar ve fonksiyon tipleri 33
BASIC anahtar-sözcüklerinin açıklamaları (alfabetik sıraya göre) 36
Commodore 64 klavyesi ve özellikleri 87
Ekran editörü 89

3. COMMODORE 64'TE GRAFİKLERİN PROGRAMLANMASI 93
Grafiklere genel bakış 94
  Karakter gösterim modları 94
  Bit harita modları 94
  Yaratıklar 94
Grafik konumları 94
  Video küme seçimi 95
  Ekran belleği 96
  Renk belleği 97
  Karakter belleği 97

Standard karakter modu 99
Karakter tanımları 100
Programlanabilir karakterler 100
Çok-renkli grafikler 106
Çok-renkli mod biti 107
Geliştirilmiş zemin-rengi modu 111
Bit haritalama grafikleri 112
Bit haritalama 112
Standard yüksek çözünürlük bit haritalama modu 112
Nasıl çalışır? 113
Çok renkli bit haritalama modu 117
Düz kaydırma (smooth scrolling) 118
Yaratıklar (sprites) 120
Yaratık tanımlama 121
Yaratık göstergeçleri 122
Yaratık etkinleştirme 123
Yaratık devre dışı bırakma 123
Renkler 123
Çok-renkli mod 124
Bir yaratığı çok-renkli modda oluşturma 124
Yaratık genişletme 125
Yaratık konumlandırma 126
Dikey konumlandırma 127
Yatay konumlandırma 127
Yaratık konumlandırma özeti 129
Yaratık görüntüleme öncelikleri 130
Çarpışma 130
Yaratık ile yaratık çarpışmaları 131
Yaratık ile veri çarpışmaları 131
Diğer grafik özellikleri 136
Ekran temizleme 136
Tarama kaydı 136
Kesinti durum kaydı 136
Önerilen ekran ve karakter renk kombinasyonları 137
Yaratık programlama-bir başka bakış 138
BASIC'le yaratıklar oluşturabilirsiniz-kısa bir program 138
Yaratık programlarınızı kısaltabilirsiniz 141
Yaratıkların ekran üzerine yerleştirilmesi 142
Yaratık öncelikleri 146
Yaratıkları çizelim 146
Bir yaratığın aşama aşama oluşturulması 147
Yaratıkların hareketlendirilmesi 150
Düşey kayma (vertical scrolling) 151
Bir yaratık programı örneği-dans eden fare 151
Yaratık oluşturma tablosu 157
Yaratık oluşturma notları 157

4. COMMODORE 64'TE SESİ VE MÜZİĞİ PROGRAMLAMA 163
Giriş 164
Volüm (ses düzeyi) kontrolü 165
Ses dalgalarının frekansları 166
Birden fazla sesin kullanımı 167
Çok sesliliğin kontrolü 170
Dalga biçimi değişimleri 171
Dalga biçimlerini anlama 173
Zarf üreticisi 174
Filtreleme 177
İleri teknikler 179
Senkronizasyon ve ring modülasyonu 183

5. BASIC'TEN MAKİNE DİLİNE 187
Makine dili nedir? 189
Makine kodları neye benzer? 189
Commodore 64'ün bellek haritası 190
6510 mikroişlemci içerisindeki kayıtlar 191
Makine dili ile nasıl program yazabilirsiniz? 192
64MON 193
Onaltılı sayılar 193
İlk makine dili komutunuz 195
İlk programınızı yazarken 196
Adresleme modları 197
0'ıncı sayfa 197
Yığın 198
İndeksli adresleme 198
Dolaylı adresleme 198
Dolaylı indeksli adresleme 198
İndeksli dolaylı adresleme 199
Dallanma ve koşullu komutlar 199
Altprogramlar 201
Yeni başlayanlar için yararlı ipuçları 202
Büyük bir işe başlarken 202
MCS6510 mikroişlemcisinin komut kümesi alfabetik sıra ile 203
Adresleme modları ve işletim zamanları 220
Commodore 64'ün bellek yönetimi 224
KERNAL 231
KERNAL'ın başlangıç aktiviteleri 232
KERNAL nasıl kullanılır? 232
Kullanıcı tarafından çağrılabilen KERNAL rutinleri 234
Hata kodları 259
BASIC'ten makine dili kullanımı 260
Makine dili rutinleri nereye konulur? 261
Makine dili nasıl girilir 262
Commodore 64 bellek haritası 263
Commodore 64 giriş/çıkış atamaları 268

6. GİRİŞ/ÇIKIŞ REHBERİ 277
Giriş 279
TV için çıkış 279
Diğer cihazlar için çıkış 280
Yazıcı için çıkış 281
Modem için çıkış 282
Teyp ünitesi ile çalışma 283
Floppy disketlerde veri saklama 284
Oyun portları........ 284
Kürekler 287
Işık kalemi 289
RS-232 arabirimi açıklaması 289
Genel taslak 289
RS-232 kanalının açılması 290
RS-232 kanalından veri almak 293
RS-232 kanalına veri gönderilmesi 293
RS-232 veri kanalının kapatılması 294
Örnek BASIC programları 295
Alıcı/verici deposu taban adresi göstergecileri 296
Sıfırıncı-sayfanın bellekteki yerleri ve RS-232 arabirimi kullanımı 297
Sıfırıncı-sayfa olmayan bellek yerleri ve kullanımı 297
Kullanıcı portu 297
Port ucu açıklaması 298
Seri bağlantı 299
Seri bağlantı çıkış uçları 301
Genişleme portu 303
Z 80 mikroişlemci kartuşu 305
Commodore CP/M® kullanımı 305
Commodore CP/M® çalıştırması 306

EKLER 309
A - BASIC sözcükleri için kısaltmalar 311
B - Ekran gösterim kodları 313
C - ASCII ve CHR$ kodları 315
D - Ekran ve renk bellek haritaları 317
E - Müzik nota değerleri 319
F - Bibliyografya 323
G - VIC-II çipi kayıt haritası 326
H - Matematiksel fonksiyonların türetilmesi 328
I - Giriş/çıkış cihazları için uç çıkışları 329
J - Standart BASIC programların Commodore 64 BASIC'ine çevrilmesi... 332
K - Hata mesajları 333
L - 6510 mikroişlemcinin özellikleri 335
M - 6526 karmaşık ara yüz adaptörü (CIA) çip özellikleri 350
N - 6566/6567 (VIC-II) çip özellikleri 365
O - 6581 ses ara yüz cihazı (SID) çip özellikleri 379
P - COMMODORE 64 ELEKTRONİK DEVRE ŞEMASI 397

R - Mini Sözlük 399
COMMODORE 64 HIZLI REFERANS KARTI 400
PROGRAM ÖRNEKLERİ 401
NOTLAR 407

BAŞLARKEN

Commodore 64 programcının başvuru kılavuzu, Commodore 64'ünüzün yerleşik özelliklerini en üst düzeye çıkarmak isteyenler için bir çalışma aracı ve başvuru kaynağı olarak geliştirilmiştir. En basit örnekten en karmaşığa kadar. Commodore 64 programcının başvuru kılavuzu, BASIC programcılığına yeni başlayanlardan 6502 makine dilinde deneyimli profesyonellere kadar herkesin kendi yaratıcı programını geliştirmek için bilgi alabileceği şekilde tasarlanmıştır. Aynı zamanda bu kitap size Commodore 64'ünüzün gerçekten ne kadar akıllı olduğunu gösteriyor.

Bu başvuru kılavuzu, BASIC programlama dilini veya 6502 makine dilini öğretmek için tasarlanmamıştır. Bununla birlikte, kapsamlı bir terimler sözlüğü ve kitaptaki bölümlerin çoğunda "yarı öğretici" bir yaklaşım vardır. BASIC ve onu programlamak için, nasıl kullanacağınız konusunda, henüz bir çalışma bilgisine sahip değilseniz, bilgisayarınızla birlikte gelen Commodore 64 kullanıcı kılavuzunu incelemenizi öneririz. Commodore 64 programcının başvuru kılavuzu size BASIC programlama diline okuması kolay bir giriş sağlar. BASIC'i nasıl kullanacağınızı anlamakta hala zorluk çekiyorsanız, bu kitabın arkasına (veya kullanıcı kılavuzundaki Ek: N'ye) dönün ve kaynakçaya bakın.

Commodore 64 programcının başvuru kılavuzu tam olarak budur; referans. Çoğu referans kitap gibi, bilgiyi yaratıcı bir şekilde uygulama yeteneğiniz de konu hakkında ne kadar bilgi sahibi olduğunuza bağlıdır. Başka bir deyişle, acemi bir programcıysanız, mevcut programlama bilginizi genişletene kadar bu kitaptaki tüm bilgi ve teknikleri kullanamayacaksınız.

Hadi şimdi bilgisayarımızın güç düğmesini açalım.



Bu kitapla yapabileceğiniz şey, programcılık yapısının açıklandığı, okunması kolay, sade Türkçe ile yazılmış önemli miktarda değerli programlama referans bilgisine ulaşmaktır. Öte yandan, programlama uzmanı Commodore 64'ün yeteneklerini etkin bir şekilde kullanmak için gereken tüm bilgileri bulacaktır.

KAPSAMI NEDİR?

- Eksiksiz "BASIC sözlüğümüz", Commodore BASIC dil komutlarını, deyimlerini ve alfabetik sırayla listelenmiş işlevlerini içerir. Tüm kelimeleri ve kısaltmalarını içeren bir "hızlı liste" oluşturduk. Bunu, nasıl çalıştıklarını göstermek için örnek BASIC programları ile birlikte her kelimenin daha ayrıntılı bir tanımını içeren bir bölüm takip eder. BASIC programlarıyla makine dilini kullanmak için bir girişe ihtiyacınız varsa, uzman olmayan genel bakışımız başlamanıza yardımcı olacaktır.

- Tüm Commodore bilgisayarlarının güçlü bir özelliği KERNAL olarak adlandırılır. KERNAL, Commodore bilgisayarınızda yazdığınız tüm programların yarın da kullanılabilmesine yardımcı olur.

- Giriş/Çıkış Programlama bölümü size bilgisayarınızı sonuna kadar kullanma fırsatı verir. Işıklı kalemler ve oyun kolundan disk sürücülerine, yazıcılara ve modem adı verilen telekomünikasyon cihazlarına kadar her şeyin nasıl bağlanacağını ve kullanılacağını açıklar.

- Mikrobilgisayar endüstrisindeki en ayrıntılı ve gelişmiş animasyonlu resimler için YARATIKLAR, programlanabilir karakterler ve yüksek çözünürlüklü grafikler dünyasını keşfedebilirsiniz.

- Ayrıca herhangi bir kişisel bilgisayarda bulunan en iyi yerleşik sentezleyici ile müzik sentezi dünyasına girebilir ve kendi şarkılarınızı ve ses efektlerinizi oluşturabilirsiniz.

- Deneyimli bir programcıysanız, yazılım yükleme dili bölümü size Commodore 64'ün CP/M* ve yüksek seviyeli dilleri çalıştırma yeteneği hakkında bilgi verir.

Bu, BASIC'e ektir. Commodore 64 programcının başvuru kılavuzu size yardımcı olacak kullanışlı bir araç olarak düşünün ve önünüzdeki programlama saatlerinin tadını çıkarın.

*CP/M, Digital Research, Inc.'in tescilli ticari markasıdır.

BU REHBER NASIL KULLANILIR?

Bu kılavuz boyunca, BASIC komutlarının veya deyimlerinin söz dizimini (programlama cümle yapısı) tanımlamak ve her bir BASIC anahtar sözcüğünün hem gerekli hem de isteğe bağlı kısımlarını göstermek için belirli geleneksel gösterimler kullanılmıştır. Deyim söz dizimini yorumlamak için kullanılacak kurallar aşağıdaki gibidir:

1. BASIC anahtar sözcükleri büyük harflerle gösterilir. Ekstrede gösterilen, girilen ve tam olarak gösterildiği gibi yazıldığı yerde görünmelidirler.
2. Tırnak işaretleri ("") içinde gösterilen öğeler, girmeniz gereken değişken verileri belirtir. Hem tırnak işaretleri hem de tırnak içindeki veriler, her ifadede gösterildiği yerde görünmelidir.
3. Köşeli parantezler ([]) içindeki öğeler, isteğe bağlı bir ifade parametresini belirtir. Parametre, ifadeleriniz için bir sınırlama veya ek niteleyicidir. İsteğe bağlı bir parametre kullanıyorsanız, bu isteğe bağlı parametre için verileri sağlamanız gerekir. Ek olarak, üç nokta (…) isteğe bağlı bir öğenin bir programlama satırının izin verdiği kadar tekrarlanabileceğini gösterir.
4. Köşeli parantez ([]) içindeki bir öğenin ALT ÇİZGİLİ ise, isteğe bağlı parametrelerde bu belirli karakterleri kullanmanız ZORUNLUDUR ve bunlar da tam olarak gösterildiği gibi yazılmalıdır.
5. Açısal parantezler (<>) içindeki öğeler, sağladığınız değişken verileri gösterir. Bölüm işareti (/), birbirini dışlayan iki seçenek arasında bir seçim yapmanız gerektiğini belirtir.

Yazım formatı örneği:

OPEN <dosya-no>, <cihaz> [, <adres>], "[<sürücü>: <dosya-ismi>] [, <mod>]"

```
10 OPEN 2,8,6,"0:CARI KARTI,S,W"
20 OPEN 1,1,2,"STOK KARTI"
30 OPEN 3,4
```

Pratik bir durumda söz dizimi kurallarını gerçekten uyguladığınızda, ifadelerinizdeki parametrelerin sırası, söz dizimi örneklerinde gösterilen sıra ile tam olarak aynı olmayabilir. Örnekler, olası her diziyi göstermek için değildir. Tüm gerekli ve isteğe bağlı parametreleri sunmaları amaçlanmıştır.

Bu kitaptaki programlama örnekleri, okunabilirlik açısından sözcükleri ve işleçleri ayıran boşluklarla gösterilmiştir. Normalde, BASIC, sözcükleri dışarıda bırakmak size belirsiz veya yanlış bir söz dizimi vermedikçe, sözcükler arasında boşluk gerektirmez.

Aşağıdaki bölümlerde çeşitli ifade parametreleri için kullanılan sembollerin bazı örnekleri ve açıklamaları aşağıda gösterilmiştir. Liste her olasılığı göstermek için değil, söz dizimi örneklerinin nasıl sunulduğunu daha iyi anlamanızı sağlamak içindir.

SEMBOL	ÖRNEK	AÇIKLAMA
<dosya no>	50	Mantıksal bir dosya numarası
<cihaz>	4	Bir donanım cihaz numarası
<adres>	15	Bir seri veri yolu ikincil aygıt adresi
<sürücü>	8	Bir fiziksel disk sürücüsü numarası
<dosya adı>	"TEST.PRG"	Bir verinin veya program dosyasının adı
<sabit veri>	"ABCEFG"	Programcı tarafından sağlanan veriler
<değişken>	X145	Herhangi bir BASIC veri değişken adı veya sabiti
<yazınsal dizi>	ABCD	Bir yazınsal dizi değişkenin kullanılması gerekir
<sayı>	12345	Bir sayı değişkenin kullanılması gerekir
<satır no>	100	Program satır numarası
<sayısal>	1,5E4	Bir tam sayı veya ondalık sayı değişkeni

COMMODORE 64 UYGULAMA KILAVUZU

Bir bilgisayar satın almayı ilk düşündüğünüzde, muhtemelen kendinize, "Artık bir bilgisayar satın almaya gücüm yettiğine göre, bir tane aldığımda onunla ne yapabilirim?" diye sormuşsunuzdur.

Commodore 64'ünüzle ilgili harika olan şey, ona istediğinizi yaptıra bilmenizdir! Ev ve iş yeri bütçe ihtiyaçlarını hesaplamasını ve takip etmesini sağlayabilirsiniz. Kelime işlem için kullanabilirsiniz. Atari oyunları oynamasını sağlayabilirsiniz. Şarkı söylemesini sağlayabilirsiniz. Hatta kendi çizgi filmlerinizi ve daha fazlasını oluşturabilirsiniz. Bir Commodore 64'e sahip olmanın en iyi yanı, aşağıda listelenenlerden sadece birini yapsa bile, ödediğiniz fiyata değecek olmasıdır. Ancak 64, eksiksiz bir bilgisayardır ve listelenen HER ŞEYİ ve sonra bazılarını yapar!

Bu arada, buradaki her şeye ek olarak, yerel bir Commodore Kullanıcı Kulübüne kaydolarak, COMMODORE ve POWER/PLAY dergilerine abone olarak ve CompuServe™ üzerinde COMMODORE BİLGİ AĞINA katılarak birçok yaratıcı ve pratik fikir edinebilirsiniz

UYGULAMA	YORUM / ŞARTLAR
AKSİYON OYUNLARI	Omega Race, Gorf, Wizard of Wor gibi gerçek Bally Midway arcade oyunlarının yanı sıra Math Teacher 1, Home Babysitter ve Commodore Artist gibi “oyna ve öğren” oyunlarına da sahip olabilirsiniz.
REKLAM VE TİCARET	COMMODORE 64'ünüzü bir TV'ye bağlayın, yanıp sönen, animasyonlu ve müzikli bir mesajla bir mağaza penceresine koyun ve harika bir mağaza ekranı satın alma noktasına sahip olursunuz.
ANİMASYON	Commodore 64'ün yaratık grafikleri, şekillerin birbirinin önünde veya arkasında hareket edebilmesi için 8 farklı seviye ile gerçek çizgi filmler oluşturmanıza olanak tanır.
BEBEK BAKICILIĞI	COMMODORE 64 HOME BABYSISTER kartuşu, en küçük çocuğunuzu saatlerce oyalayabilir ve aynı zamanda alfabe/klavye tanımayı öğretebilir. Ayrıca özel öğrenme kavramlarını ve ilişkilerini öğretir.
BASIC PROGRAMLAMA	COMMODORE 64 KULLANICI KILAVUZU ve KENDİNİZİ ÖĞRETİN PROGRAMLAMA kitap ve bant seriniz mükemmel bir başlangıç noktası sunar.
İŞ TABLOSU	Commodore 64, herhangi bir kişisel bilgisayar için mevcut olan en güçlü kelime işlemci ve en büyük elektronik tablo dahil olmak üzere “Easy” iş yardımları serisini sunar.
İLETİŞİM	Bilgisayar “ağlarının” büyüleyici dünyasına girin. Commodore 64'ünüze bir VIC-MODEM bağlarsanız, dünyanın her yerindeki diğer bilgisayar sahipleriyle iletişim kurabilirsiniz. Bununla da kalmayıp CompuServe™ üzerinden COMMODORE BİLGİ AĞINA üye olarak tüm Commodore ürünleri, finansal bilgiler, evden alışveriş hizmetleri hakkında en son haberleri ve güncellemeleri alabilir, hatta kurduğunuz bilgi sistemleri üzerinden arkadaşlarınızla oyun oynamaya katılabilirsiniz.

ŞARKILAR YAPMAK

Commodore 64, herhangi bir bilgisayarda bulunan en gelişmiş yerleşik müzik sentezleyici ile donatılmıştır. Tamamen programlanabilir üç sese, dokuz tam müzik oktavına ve dört kontrol edilebilir dalga biçimine sahiptir. Her türlü müzik ve ses efektini yaratmanıza veya yeniden üretmenize yardımcı olacak Commodore Müzik Kartuşlarını ve Commodore Müzik kitaplarını arayın.

CP/M*

Commodore, bir CP/M* eklentisi ve yüklemesi kolay bir kartuş aracılığıyla yazılıma erişim sunar.

YETENEK EĞİTİMİ

El/Göz koordinasyonu ve el becerisi, “Jupiter Lander” ve gece sürüş simülasyonu dahil olmak üzere çeşitli Commodore oyunları tarafından desteklenir.

EĞİTİM

Bilgisayarla çalışmak başlı başına bir eğitim olmakla birlikte, Commodore Eğitim Kaynak Kitabı bilgisayarların eğitsel kullanımları hakkında genel bilgiler içermektedir. Ayrıca müzikten matematiğe, sanattan astronomiye kadar her şeyi öğretmek için tasarlanmış çeşitli öğrenme kartuşlarımız var.

YABANCI DİL

Commodore 64 programlanabilir karakter seti, standart karakter setini kullanıcı tanımlı yabancı dil karakterleriyle değiştirmenizi sağlar. Yukarıda bahsedilen yaratık grafiğe ek olarak.

GRAFİK VE SANAT

Commodore 64 yüksek çözünürlük, çok-renkli grafik çizimi, programlanabilir karakterler ve tüm farklı grafik ve karakter görüntüleme modlarının kombinasyonlarını sunar.

CİHAZ KONTROLÜ

Commodore 64'ünüz, çeşitli özel endüstriyel uygulamalarla kullanım için bir seri bağlantı noktasına, RS-232 bağlantı noktasına ve bir kullanıcı bağlantı noktasına sahiptir. İsteğe bağlı ekstra bir IEEE/488 kartuşu da mevcuttur.

DERGİLER VE BUROŞÜRLER

Commodore 64 yakında mevcut çoğu "yüksek fiyatlı" kelime işlemcinin kalite ve esnekliklerine uyan veya aşan, istisnai bir kelime işlemci sistemi sunacak. Elbette bu bilgileri 1541 Disk Sürücüsüne veya Datasette™ kaydediciye kaydedebilir ve bir VIC-PRINTER veya PLOTTER kullanarak yazdırabilirsiniz.

---

*CP/M, Digital Research, Inc.'in tescilli ticari markasıdır.

IŞIKLI KALEM KONTROLÜ	Bir ışık kalemi kullanımını gerektiren uygulamalar, Commodore 64 oyun bağlantı noktası konektörüne uyan herhangi bir ışık kalemi ile gerçekleştirilebilir.
MAKİNE KODU KONTROLÜ	Commodore 64 programcının başvuru kılavuzu, bir makine dili bölümünün yanı sıra bir BASIC makine kodu ara yüzü bölümü içerir. Daha derinlemesine çalışma için bir bibliyografya bile var.
BORDRO VE FORM BASKILARI	Commodore 64, çeşitli giriş tipi iş uygulamalarını işlemek üzere programlanabilir. Commodore 64 "iş formu" grafikleriyle birleştirilmiş büyük/ küçük harfler, daha sonra yazıcınızdan yazdırılabilecek formlar tasarlamanızı kolaylaştırır.
BASKI	Commodore 64, çeşitli nokta vuruşlu ve mektup kalitesinde yazıcıların yanı sıra çizicilerle ara yüz oluşturur.
TARİFLER	En sevdiğiniz tarifleri Commodore 64'ünüze ve onun disk veya kaset saklama ünitesine kaydedebilir ve en çok ihtiyacınız olduğunda genellikle kaybolan dağınık tarif kartlarına olan ihtiyacınıza son verebilirsiniz.
SİMÜLASYONLAR	Bilgisayar simülasyonları, minimum risk ve maliyetle tehlikeli veya pahalı deneyler yapmanızı sağlar.
SPOR VERİLERİ	Source™ ve CompuServe™, Commodore 64 ve bir VIC-MODEM kullanarak alabileceğiniz spor bilgileri sunar.
STOK TEKLİFLERİ	Bir VIC-MODEM ve uygun ağ hizmetlerinden herhangi birine abonelik ile Commodore 64' ünüz kendi özel hisse senedi kaydınız olur.

Bunlar sizin ve Commodore 64'ünüz için birçok uygulamadan sadece birkaçı. Gördüğünüz gibi, iş veya oyun için, evde, okulda veya ofiste, Commodore 64'ünüz size hemen her ihtiyaç için pratik bir çözüm sunuyor.

Commodore, kullanıcılara verdiğimiz desteğin yalnızca bir Commodore bilgisayarı satın aldığında başladığını bilmenizi ister. Bu nedenle, dünyanın dört bir yanından değerli girdiler içeren "iki yönlü" bir bilgisayar bilgi ağı oluşturduk.

Ayrıca, dünya çapında Commodore Kullanıcı Kulüplerinin büyümesini yürekten teşvik ediyor ve destekliyoruz. Başlangıç seviyesinden en ileri seviyeye kadar her Commodore bilgisayar sahibi için mükemmel bir bilgi kaynağıdır. Aşağıda daha ayrıntılı olarak açıklanan dergiler ve ağ, bölgenizdeki Kullanıcılar Kulübüne nasıl katılacağınız konusunda en güncel bilgilere sahiptir. Son olarak, yerel Commodore bayiniz yararlı bir Commodore destek ve bilgi kaynağıdır.

COMMODORE BİLGİ AĞI

Geleceğin dergisi burada. POWER/PLAY ve COMMODORE dergilerine aboneliğinizi tamamlamak ve geliştirmek için, "kağıtsız dergimiz" COMMODORE INFORMATION NETWORK, Commodore bilgisayarınızı ve modeminizi kullanarak artık telefonda mevcuttur.

Bilgisayar kulübümüze katılın, bir bilgisayar sorunuyla ilgili yardım alın, diğer Commodore arkadaşlarınızla "konuşun" veya yeni ürünler, yazılımlar ve eğitim kaynakları hakkında en güncel bilgileri alın. Yakında, POWER/PLAY veya COMMODORE'da bulduğunuz program listelerini doğrudan COMMODORE BİLGİ AĞI'ndan indirerek yazma zahmetinden kendinizi kurtarabileceksiniz (1983'ün başlarında yeni bir kullanıcı hizmeti planlandı). En iyi yanı, cevapların çoğunun, siz daha soruları sormadan önce orada hazır olmasıdır. (Bu nasıl bir hizmet kavrayabiliyor musunuz?)

Elektronik dergimizi aramak için sadece bir modeme ve ülkenin en büyük telekomünikasyon ağlarından biri olan CompuServe™ aboneliğine ihtiyacınız var. (İşinizi kolaylaştırmak için Commodore, her VIC-MODEM paketinde CompuServe™ için ücretsiz bir yıllık abonelik içerir.) CompuServe™ veri bankası için yerel numaranızı çevirmeniz ve telefonunuzu modeme bağlamanız yeterlidir. Ekranınızda CompuServe™ video metni göründüğünde, bilgisayar klavyenizde G CBM yazın. COMMODORE BİLGİ AĞI'nın içindekiler tablosu veya menü ekranınızda göründüğünde, on altı bölümümüzden birini seçin, rahatınıza bakın ve diğer dergilerin hakkında yazdığı kağıtsız derginin keyfini çıkarın.

Daha fazla bilgi için Commodore bayinizi ziyaret edin veya 800 848 8990 (Ohio'da, 614 457 8600) numaralı telefondan CompuServe™ müşteri hizmetleri ile iletişime geçin.

COMMODORE BİLGİ AĞI

Main Menu Description	Commodore Dealers
Direct Access Codes	Educational Resources
Special Commands	User Groups
User Questions	Descriptions
Public Bulletin Board	Questions and Answers
Magazines and Newsletters	Software Tips
Products Announced	Technical Tips
Commodore News Direct	Directory Descriptions

BÖLÜM 1

BASIC PROGRAMLAMA KURALLARI

- Giriş
- Ekran gösterim kodları (BASIC'in temel karakter seti)
- Sabit ve değişkenleri programlama
- İfadeler ve operatörler
- Programlama teknikleri

GİRİŞ

Bu bölüm, BASIC'in verileri nasıl depoladığı ve yönettiği hakkında bilgi veriyor. Konular şunları içerir:

1. Commodore 64'te kullanılan karakter kümesinin yanı sıra işletim sistemi bileşenleri ve işlevleri hakkında kısa bir açıklama.

2. Sabitlerin ve değişkenlerin oluşumu. Ne tür değişkenler vardır. Sabitler ve değişkenler bellekte nasıl saklanır.

3. Aritmetik hesaplamalara, bağıntı testlerine, yazınsal dizilere ve mantıksal işlemlere (logical operations) ilişkin kuralların yanı sıra BASIC'te karma veri türleri ile bunların birbirlerine çevrilmesi için gerekli kurallar ve bunların kullanıldığı ifadelerin (expres-sions) oluşturulması.

EKRAN GÖSTERİM KODLARI (BASIC TEMEL KARAKTER SETİ)
İŞLETİM SİSTEMİ (OS)

İşletim Sistemi Salt Okunur Bellek (ROM) yongalarında bulunur ve üç ayrı, ancak birbiriyle ilişkili program modülünün birleşimidir:

1. BASIC Yorumlayıcı
2. KERNAL
3. Ekran Editörü

1. BASIC Yorumlayıcısı, BASIC deyim söz dizimini analiz etmekten ve gerekli hesaplamaları ve/veya veri işlemeyi gerçekleştirmekten sorumludur. BASIC Yorumlayıcı, özel anlamları olan 65 "anahtar kelimeden" oluşan bir kelime dağarcığına sahiptir. Büyük ve küçük harf alfabesi ve 0-9 arasındaki rakamlar hem anahtar kelimeleri hem de değişken adlarını yapmak için kullanılır. Belirli noktalama işaretleri ve özel sembollerin de Yorumlayıcı için bir anlamları vardır. Tablo1-1 özel karakterleri ve kullanımlarını listeler.

2. KERNAL, sistemdeki kesme düzeyindeki işlemlerin çoğunu yönetir (kesme düzeyinde işlemeyle ilgili ayrıntılar için bkz. Bölüm 5). KERNAL ayrıca verilerin gerçek giriş ve çıkışını da yapar.

3. Ekran Düzenleyici, video ekranının (televizyon seti) çıkışını ve BASIC program metninin düzenlenmesini kontrol eder. Ek olarak, ekran düzenleyici klavye girişini keser, böylece girilen karakterler üzerinde hemen harekete mi geçileceğine veya BASIC yorumlayıcısına mı aktarılacağına karar verebilir.

Tablo 1-1. CBM BASIC karakter seti

KARAKTER	İSİM	AÇIKLAMA
	BOŞLUK	Anahtar sözcükleri ve değişken adlarını ayırır.
;	NOKTALI VİRGÜL	PRINT yönergesinde, ekrana ya da başka bir cihaza gönderilen karakterler arasında boşluk bırakılmayacağını belirtir.
=	EŞİT İŞARETİ	a) Değişkenlerin değerini belirlemede. b) Aritmetik eşitliği belirtmekte kullanılır.
+	ARTI İŞARETİ	Aritmetik toplama veya dizi birleştirme. (Birleştirme: bir zincirde birbirine bağlama).
−	EKSİ İŞARETİ	Aritmetik çıkarma, negatifleştirme (-1).
*	ÇARPI İŞARETİ	Aritmetik çarpma.
/	BÖLÜM İŞARETİ	Aritmetik bölme.
↑	ÜS ALMA	Aritmetik üs alma.
(	SOL PARANTEZ	Aritmetik parantez ve fonksiyonlar.
)	SAĞ PARANTEZ	Aritmetik parantez ve fonksiyonlar.
%	YÜZDE	Değişken adını bir tamsayı olarak bildirir.
#	SAYI	Giriş/çıkış yönergelerinde dosya no belirtir. (GET#, PRINT#, INPUT# gibi).
$	DOLAR İŞARETİ	Değişken adını yazınsal dizi olarak belirler.
,	VİRGÜL	a) Komut parametrelerini ayırır. b) PRINT yönergesinde karakterler arasında tablolama yapılmasını sağlar.
.	NOKTA	Ondalık sayı gösterir.
"	TIRNAK İŞARETİ	Yazınsal dizi sabitlerini oluşturur.
:	İKİ NOKTA	Bir satırda birden fazla BASIC deyimini ayırır.
?	SORU İŞARETİ	PRINT anahtar kelimesinin kısaltılması.
<	KÜÇÜKTÜR	Aritmetik bağıntı.
>	BÜYÜKTÜR	Aritmetik bağıntı.
π	Pİ	Sabit sayı 3.141592654

İşletim Sistemi size iki BASIC çalışma modu sunar:

1. Doğrudan mod
2. Program mod

1. Doğrudan mod kullandığınızda, BASIC deyimlerinin önünde satır numaraları bulunmaz. **RETURN** tuşuna her basıldığında yürütülürler.

2. Program modu, programları çalıştırmak için kullandığınız moddur. Program modunu kullanırken, tüm BASIC ifadelerinizin önünde satır numarası olmalıdır. Programınızın bir satırında birden fazla BASIC ifadesi olabilir. Mantıksal bir ekran satırı 80 karakter ile sınırlıdır. 80 karakter sınırını aşacaksanız, satıra sığmayan BASIC ifadesinin tamamını yeni bir satır numarasıyla başlamanız gerekir

NOT: Yeni bir programa başlamadan önce daima NEW yazın ve RETURN tuşuna basın.

Commodore 64, klavyeden veya programınızdan kullanabileceğiniz iki ayrı karakter setine sahiptir.

SET 1'de SHIFT tuşuna basılmadan büyük harfler ve 0-9 arası sayılar kullanılabilir. Yazarken SHIFT tuşunu basılı tutarsanız, tuşların ön tarafındaki SAĞ taraftaki grafik karakterleri kullanılır. Yazarken C= tuşuna basılı tutarsanız, tuşun ön yüzünün SOL tarafındaki grafik karakterleri kullanılır. Tuşun önünde grafik sembolleri olmayan herhangi bir karakteri yazarken SHIFT tuşunu basılı tutmak, tuşun en üst kısmındaki sembolü size verir.

SET 2'de SHIFT tuşuna basılmadan küçük harfler ve 0-9 arası sayılar kullanılabilir. Büyük harfler ise, yazarken SHIFT tuşuna basılı tutularak yazılır. Yine tuşların ön yüzündeki SOL taraftaki grafik semboller C= tuşuna basılarak görüntülenirken, grafik karakterleri olmayan herhangi bir tuşun en üst kısmındaki semboller, yazarken SHIFT tuşunu basılı tuttuğunuzda elde edilir.

Bir karakter kümesinden diğerine geçmek için C= ve SHIFT tuşuna birlikte basın.

Birinci karakter setten ikinci karakter sete geçmek.

```
PRINT CHR$(14)
```

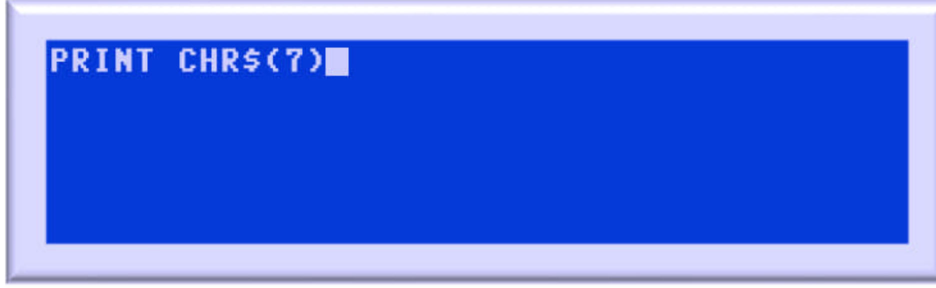
İkinci karakter setten birinci karakter sete geçmek.

```
PRINT CHR$(142)
```

Karakter setini kilitlemek ve C= SHIFT tuşunu işlemez hale getirmek.

```
PRINT CHR$(8)
```

Karakter setini açmak ve C= SHIFT tuşunu işler hale getirmek.



SABİT VE DEĞİŞKEN DEĞERLERİN PROGRAMLANMASI
TAM SAYI, ONDALIK SAYI VE YAZINSAL DİZİ SABİTLERİ

Sabitler, BASIC ifadelerinize koyduğunuz veri değerleridir. BASIC, deyim yürütme sırasında verileri temsil etmek için bu değerleri kullanır. CBM BASIC, üç tür sabiti tanıyabilir ve değiştirebilir:

1. Tam sayılar
2. Ondalık sayılar
3. Yazınsal diziler

1. Tam sayılar:

Ondalık sayı olmayan tüm sayıları kapsar. Tam sayı sabitleri -32768 ile +32767 arasında olmalıdır. Tam sayı sabitlerinde, rakamlar arasında nokta ya da virgül yoktur. Eğer sol tarafla (+) işareti yoksa, sayı pozitif olarak kabul edilir. Sayıdan önce gelen sıfırlar anlamsızdır. Bunlar programınızı yavaşlatacağından ve bellekte yer kaplayacaklarından ötürü kullanılmamalı; ancak bunları kullanmak herhangi bir yanlışa yol açmaz. Tam sayılar bellekte iki baytlık, ikili sayı olarak kaydedilirler. Tam sayı sabitlerine şu örnekleri verebiliriz.

-12
8765
-32768
+44
0
-32767

NOT: Rakamlar arasında kesinlikle virgül yoktur. Örneğin 32,000 her zaman 32000 olarak yazılır. Virgül koyduğunuz zaman **?SYNTAX ERROR** (yazım hatası) BASIC hata mesajını alırsınız.

2. Ondalık sayılar:

Ondalık Sayı, sabitleri negatif ya da pozitif olabilirler. Ondalık kısım bir noktayla gösterilir. Tekrar hatırlatalım; rakamlar arasında virgül hiçbir zaman kullanılmaz. Başında (+) işareti bulunmayan sayılar Commodore 64 tarafından pozitif olarak kabul edilirler. Nokta kullanmadığınız durumda ise bilgisayar noktanın, sayının son rakamından sonra geldiğini varsayacaktır. Tam sayılarda olduğu gibi bunlarda da tam sayı kısmının başındaki sıfırlar anlamsızdır. Ondalık sayı sabitleri şu şekilde kullanılırlar:

1. Basit sayı
2. Bilimsel yazım

Ondalık sayı sabitleri ekranınızda 9 rakamlı sayılara kadar yazılabilirler. Bu rakamlar -999999999 ve +999999999 arasındaki değerleri alabilirler. 9 basamaktan fazla yazdığınız zaman, sayı onuncu rakama bağlı olarak yuvarlanır. Onuncu sayı 5'e eşit ya da büyük ise sayı bir üst sayıya, 5'ten küçük ise bir önceki sayıya yuvarlanır. Bu bazı sayıların son toplam değerleri için önemli olacaktır.

Ondalık sayılar bellekte 5 bayt kullanılarak kaydedilirler ve hesaplamalarda on basamağa kadar doğrulukla işlenirler. Ancak sayılar yazılırken yine 9 rakama yuvarlanır. Basit ondalık sayılara şu örnekleri verebiliriz.

```
    1.23
-.998877
 +3.1459
.7777777
   -333.
     .01
```

0.01'den küçük ya da 999999999.'dan büyük olan sayılar bilimsel yazım ile yazılırlar. Ondalık sayılar bilimsel yazımda üç kısımdan oluşurlar.

1. Taban
2. E harfi
3. Üs (Kuvvet)

Taban, basit bir ondalık sayıdır. E harfi ise, sayıyı üslü biçimde nasıl ifade edeceğinizi gösterir. Başka bir deyişle E *10 demektir. (Örneğin:3E3=3*10↑3=3000). Üs, sayının 10'un hangi kuvveti ile çarpılacağını gösterir.

Taban ve üs, + veya - işaretlerini alabilirler. Üs aralığı, -39'dan +38'e kadardır ve tabanda ondalık işaretinin (noktanın) kaç basamak sola (-) ya da sağa (+) konacağını gösterir.

BASIC'te bilimsel yazım kullanılsa dahi, ondalık sayıların kullanım boyutları, yine de sınırlıdır. Kullanılabilecek en büyük sayı +1.70141183E+38'dir. Sonucu bu sayıdan daha büyük olan işlemler için, **?OVERFLOW ERROR** hata mesajını alırsınız. En küçük ondalık sayı ise +2.93873588E-39'dur ve bundan küçük sayılarla sonuçlanan hesaplamalarda sonuç sıfırdır. Hata mesajı verilmez. Bilimsel yazımda gösterilen ondalık sayılara şu örnekleri verebiliriz.

```
 235.988E-3           (.235988)
     2359E6       (2359000000.)
   -7.09E-12 (-.00000000000709)
-3.14159E+5          (-314159.)
```

3. Yazınsal dizi:

Yazınsal dizi sabitleri ise harfler, sayılar ve semboller gibi alfa sayısal veri gruplarıdır. Klavyeden bir yazınsal dizi girerken 80 karakterlik satıra sığacak şekilde, istediğiniz kadar karakter girebilirsiniz.

Bir yazınsal dizi sabiti, herhangi bir şekilde birleştirilmiş boşlukları, harfleri, sayıları, noktalama işaretlerini ve renk veya takipçi kontrol karakterlerini içerebilir. Bir dizide kullanılmayacak tek karakter, çift tırnak işaretidir ("). Bunun nedeni (") işaretinin dizinin başlangıcını ve bitimini belirtmek için kullanılmasıdır. Bir dizi aynı zamanda boş da olabilir, yani dizide hiç karakter kullanılmayabilir.

Bir yazınsal dizide, yönerge satırının sonunda ise veya arkasından iki nokta üst üste (:) geliyorsa, dizinin sonuna tırnak işareti gerekmez. Dizi sabitlerine şu örnekleri verebiliriz:

İki tırnak (“”) boş bir yazınsal diziyi ifade eder.

```
PRINT ""
```

```
PRINT "MERHABA"
```

```
PRINT "25,000.00TL"
```

```
PRINT "ISCI SAYISI"
```

NOT: Yazınsal dizi içinde (") işaretinin içerilmesini istiyorsanız CHR$(34) kullanın.

TAM SAYI, ONDALIK SAYI VE YAZINSAL DİZİ DEĞİŞKENLERİ

Değişkenler, BASIC yönergelerindeki veri değerlerini göstermek için kullanılan isimlerdir. Bir değişkenle gösterilen değer, bir sabite eşitlenebilir ya da programda yer alan hesaplamaların sonucunu alabilir. Değişken veri, sabitlerde olduğu gibi bir tam sayı, ondalık sayı ya da yazınsal dizi olabilir. Eğer başlangıçta değişkeninize bir değer vermemişseniz, BASIC yorumlayıcısı otomatik olarak sıfıra eşit bir değişken yaratır. Ancak bunu yapabilmesi için değişkenin bir tam sayı ya da ondalık sayı değişkeni olması gerekir. Eğer yazınsal dizi değişkeni kullanıyorsanız, bu kez yorumlayıcı boş bir dizi ("") yaratacaktır.

Bir değişken ismi istediğiniz uzunlukta olabilir. CBM BASIC için sadece ilk iki karakter önemlidir. Bu da, değişkenlerin ilk iki karakterinin birbirinden kesinlikle farklı olmasını gerektirir. Değişken adları BASIC anahtar-sözcükleriyle aynı olmamalı ve BASIC anahtar-sözcüklerini, içermemelidir. Anahtar-sözcükler tüm BASIC komutlarını, yönergelerini, fonksiyon adlarını ve mantıksal operatörleri içerirler. Eğer değişkeninizin içinde anahtar-sözcük kullanırsanız ekranınızda bir **?SYNTAX ERROR** (yazım hatası) mesajı belirir. Örneğin: **SON$="NE HABER"** ifadesi programın içinde yer aldığında, içinde "**ON**" BASIC anahtar-sözcüğü bulunduğu için geçersizdir.

Değişken adlarını oluştururken tüm harfleri (alfabeyi) ve 0-9 arası rakamları kullanabilirsiniz. Ancak, ilk karakter harf olmalıdır. Veri tipini tanımlamakta kullanılan karakterler olan % ve $ işaretleri, değişken adlarının son karakterleri olarak kullanılabilirler. Yüzde işareti (%) değişkenin bir tam sayı, dolar işareti ($) ise değişkenin bir yazınsal dizi olduğunu gösterir. Eğer her ikisi de kullanılmamışsa, BASIC yorumlayıcısı değişkeni ondalık sayı değişkeni (floating-point) olarak kabul eder. Değişken adlarına, değer atamaya ve veri tiplerine, aşağıdaki örnekleri verebiliriz:

Yazınsal dizi değişkeni.

```
A$="SATIS ORANI"
```

```
MTH$="HASAN "+A$
```

Tam sayı değişkeni.

```
K%=5
```

```
CNT%=CNT%+1
```

Ondalık sayı değişkeni.

```
FP=12.5
```

```
TPLAM=FP*CNT%
```

NOT: DİKKAT! **TOPLAM** olmaz. O zaman **TO** anahtar-sözcüğü hataya neden olur.

TAM SAYI, ONDALIK SAYI VE YAZINSAL DİZİ DİZE DEĞİŞKENLERİ

Bir dize, tek bir değişken adı altında gösterilen, birbirleriyle ilişkili veri tablosu (veya listesi) olarak tanımlanabilir. Örneğin; sayılardan oluşan bir tablo bir dize olarak kabul edilebilir. Tablo içinde yer alan her bir sayı ise, dizenin "elemanları" olurlar. Dize (Array) kavramını, yazınsal dizi (String) kavramıyla karıştırmamak gerekir. İngilizce "Array" deyimi bazen "parantezli değişken" olarak da çevrilmektedir.

Dizeler birbiriyle ilişkili çok sayıdaki değişkenleri, kısaca ifade etmek için kullanılır. Şimdi sayılardan oluşan bir tabloyu düşünelim; tablonun 10 satırı, her satırda da 20 değişkeni olsun. Eğer bir dizi kullanmasaydınız, tabloda yer alan her değişken için, ayrı bir değişken adı kullanmak zorunda kalacaktınız. Oysa dize kullandığınız için tek bir isim size yeterli olacak ve dizede yer alan her eleman, dize içindeki yerleriyle tanımlanacaktır.

Dize adları tam sayı, ondalık sayı ya da yazınsal dizi veri tiplerinden herhangi birisi olabilir ve dize içinde yer alan tüm elemanlar, dizenin adıyla aynı veri tipinde olurlar. Dizeler tek boyutlu (bir liste) ya da çok boyutlu (satır ve sütunlardan oluşmuş bir tablo) olabilirler. Dizede yer alan her eleman, dize adının yanında parantez içine alınmış bir indis ile tanımlanır.

Teorik olarak bir dizenin alabileceği maksimum boyut sayısı 255 ve her boyutta bulunan eleman sayısı ise 32767 olarak belirlenmiştir. Ancak pratikte bu genişlik, verileri, kaydeden bellek kapasitesi ve/veya 80 karakterlik ekran satırı tarafından sınırlandırılır.

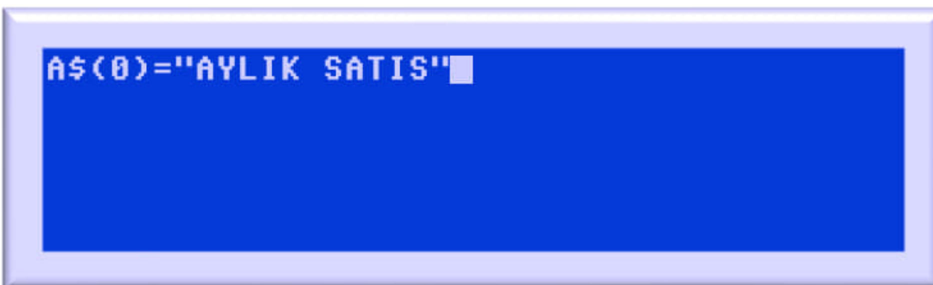
Bir dize ilk açıldığında, BASIC yorumlayıcı tarafından 11 elemanlı olarak tanımlanır. Örneğin; A(N) dizisi için bellekte. A(0)'dan A(10)'a kadar 11 yer ayrılarak 0'larla doldurulur. Böyle olmasını istemiyorsanız, dizenin genişliğini ve tipini tanımlayan BASIC DIM yönergesi kullanarak dizedeki eleman sayısını belirtmeniz gerekir. Bir dizenin iç-bellekte kaplayacağı hafıza birimini bulmak için, aşağıdaki işlemi yapın:

	5	bayt dize adı için
	+2	bayt her boyut için
	+2	bayt tam sayıların her elemanı için
VEYA	+5	bayt ondalık sayıların her elemanı için
VEYA	+3	bayt yazınsal dizilerin her elemanı için
VE	+1	bayt yazınsal dizi elemanlarının her bir karakteri için

İndisler:

Tam sayı sabitleri, değişkenler veya sonucu tam sayı verecek aritmetik ifadeler olabilirler. Dizenin her boyutu için virgülle ayrılmış farklı indisler gerekir. İndisler, sıfır ile dizenin önceden belirlenmiş maksimum boyut sayısı arasındaki herhangi bir değeri alabilirler. Bu sınırın dışındaki değerler **?BAD SUBSCRIPT** hatasına neden olurlar. Aşağıda; dize adı, değer atamaları ve veri tipleri ile ilgili örnekler göreceksiniz.

Yazınsal dizi dizesi.



```
MTH$(K%)="HAZIRAN"
```

Tam sayı dizesi.

```
G2%(X)=5
```

```
CNT%(G2%(X))=CNT%(1)-2
```

Ondalık sayı dizesi.

```
FP(12*K%)=24.8
```

```
TPL(CNT%(1))=FP↑K%
```

```
A1(C%,D%)=FP%
```

A olarak adlandırdığımız tek boyutlu bir dizenin 5'inci elamanını 0'a eşitleyelim.

```
A(5)=0
```

B olarak adlandırdığımız iki boyutlu bir dizenin 5'inci satırı ve 6'ncı sütununda yer alan elamanını 0'a eşitleyelim.

```
B(5,6)=0
```

C olarak adlandırdığımız üç boyutlu bir dizenin birinci boyutunda 5, ikinci boyutunda 6, üçüncü boyutunda 7 numaralı konumda yer alan elamanını 0'a

```
C(5,6,7)=0
```

3 boyutlu dizileri küçük küplerden oluşan büyük bir dikdörtgenler prizması gibi düşünebilirsiniz. İndisleri değiştirerek, büyük prizmayı oluşturan küçük küplerin her birine ulaşabilirsiniz.

İFADELER (EXPRESSIONS) VE OPERATÖRLER

İfadeler; sabitler, değişkenler ve/veya diziler ile aritmetik ve mantıksal işlemden oluşurlar. Bir ifade: herhangi bir tipteki, tek bir sabit, basit bir değişken veya bir dizi değişkeni olabildiği gibi, değer, aritmetik, bağıntı ve mantıksal operatörlerle birleştirilmiş birkaç sabit ve/veya değişkenden oluşabilir. Operatörlerin nasıl çalıştığı aşağıda anlatılmıştır. İfadeler iki sınıfa ayrılırlar:

1. Aritmetik
2. Yazınsal dizi

İfadeler normal olarak, işlenen dediğimiz, 2 veya daha fazla sayıdaki veri parçalarından oluşur. Her işlenen, istenilen sonucu vermek için kullanılacak olan operatör ile birbirinden ayrılır. Bu, çoğunlukla ifadenin değeri bir değişkene eşitlenerek yapılır.

Operatörler, Commodore 64'ünüzün BASIC yorumlayıcısına, değişken veya sabit veri üzerinde bir işlem yapması gerektiğini belirten özel simgelerdir. Operatörler bir veya birden fazla değişken ve/veya sabitle birlikte bir ifade oluştururlar. Commodore 64 BASIC'i; aritmetik, bağıntı ve mantıksal operatörleri tanıma yeteneğine sahiptir.

ARİTMETİK İFADELER

Aritmetik ifadeler, işleme sokulduğunda sonuçta, bir tam sayı ya da ondalık sayı verir. Toplama, çıkarma, çarpma, bölme ve kuvvet alma işlemleri için sırasıyla şu operatörler kullanılır: (+, -, *, /, ↑)

ARİTMETİK İŞLEMLER

Bir aritmetik operatör, iki yanında yer alan iki işlenen üzerinde yapılacak bir aritmetik işlemi tanımlar. Aritmetik işlemler kesirli sayılarla yapılır. Bir aritmetik işlem yapılmadan önce tüm tam sayılar kesirli sayılara çevrilir. Eğer sonucu tam sayı değişkeni olarak tanımlarsanız, elde edeceğiniz sonuç yine tam sayıya çevrilecektir.

Toplama (+):

Toplama imi, sağda bulunan işlenenin solda bulunan işlenene ekleneceğini gösterir.

```
2+2
A+B+C
X%+1
BR+10E-2
```

Çıkarma (-):

Eksi imi, sağda bulunan işlenenin solda bulunan işlenenden çıkarılacağını gösterir.

```
4-1
100-64
A-B
55-142
```

Eksi imi, (unary) negatif sayı belirtmek için de kullanılabilir. Bu, sayının sıfırdan çıkarılacağı anlamına gelir.

```
-5
-9E4
-B
4-(-2) = 4+2
```

Çarpma (*):

Asteriks imi, (*) işareti soldaki işlenenin, sağdaki işlenen ile çarpılacağını gösterir.

```
100*2
50*0
A*X1
R%*14
```

Bölme (/):

Kesme imi, işlenenin, sağındaki işlenen tarafından bölüneceğini gösterir.

10/2
6400/4
A/B
4E2/XR

Kuvvet alma (↑):

Yukarıya bakan ok imi, solda bulunan işlenenin sağda bulunan işlenen tarafından belirtildiği kadar üssünün alınacağını gösterir.

Eğer sağda bulunan işlenen 2 ise bu solda bulunan işlenenin karesini almak demektir, eğer 3'se küpünün alınacağı anlamına gelir. Üs öyle bir sayı olmalıdır ki, işlemin sonucu, geçerli sınırlar arasında bir ondalık sayı olabilsin.

2↑2	2*2'ye eşittir.
3↑3	3*3*3'e eşittir.
4↑4	4*4*4*4'e eşittir.
AB↑CD	
3↑-2	(1/3)*(1/3)'e eşittir.

BAĞINTI (RELATIONAL) OPERATÖRLERİ

Bağıntı operatörlerinin (<, =, >, <=, >=, <>) temel kullanım alanı, iki işlenenin karşılaştırılmasıdır. Ancak, bunlarla aritmetikse sonuçlar da elde edilebilir. Bağıntı ve mantık operatörleri [AND (VE), OR (VEYA), NOT (DEĞİL)] karşılaştırmalarda kullanıldıklarında, bir ifadenin aritmetiksel doğru/yanlış değerlendirilmesine dönüşürler. Eğer ifadede belirtilen ilişki doğru ise sonuç, -1, yanlış ise 0 değerini alır. İlişki operatörleri aşağıda gösterilmiştir.

<	Küçüktür
=	Eşittir
>	Büyüktür
<=	Küçüktür veya eşittir
>=	Büyüktür veya eşittir
<>	Eşit değildir

1=5-4	Doğru sonuç (-1)
14>66	Yanlış sonuç (0)
15>=15	Doğru sonuç (-1)

Bağıntı operatörleri, yazınsal dizileri karşılaştırmak için de kullanılabilirler. Karşılaştırma için alfabetik sıra temel alınır: A<B<C<D gibi. Yazınsal dizileri karşılaştırma: soldan sağa doğru, birbirine karşılık gelen karakterlerin değerlendirilmesi şeklinde yapılır. (Yazınsal dizi işlemlerine bakın).

"A"<"B"	Doğru sonuç (-1)
"X"="YY"	Yanlış sonuç (0)
BB$<>CC$	Doğru sonuç (-1)/Yanlış sonuç (0)

Karşılaştırma yaparken sayısal verilerin ancak diğer sayısal verilerle karşılaştırılacağını unutmayalım. Aynı kural yazınsal diziler için de geçerlidir. Aksi takdirde **?TYPE MISMATCH** (tip uyuşmuyor) hata mesajını alırsınız.

İki sayısal işlenen, karşılaştırılmadan önce, ondalık sayıya dönüştürülürler. Değerlendirme bundan sonra yapılır ve sonuç doğru/yanlış olarak verilir.

Bütün karşılaştırmaların sonunda işlenenin veri tipi ne olursa olsun elde edeceğiniz sonuç daima bir tam sayıdır (her ikisi de yazınsal dizi olsa dahi). Bu nedenle hesaplamalarda, işlenenlerin karşılaştırılmasında elde edilen sonuç, tek bir işlenen olarak kullanılabilir. Sonuç doğru ise -1 ya da yanlış ise 0 olacaktır. Bu sonuca, bölme işlemi dışında her şey uygulanabilir. Çünkü sıfıra bölüm geçerli değildir. Örneğin: M=A*(B>E), E=(A$-B$)+(C<D)

MANTIKSAL OPERATÖRLER

Mantıksal operatörler, AND (ve), OR (veya), NOT (değil) işlemlerinden oluşur. Başlıca işlevleri, IF (koşul) yönergelerinde, iki ifadenin ikisinin de (AND), iki ifadenin biri veya ikisinin (OR) doğru olması veya bir ifadenin doğru olmaması (NOT) koşulunu belirtmektir.

Mantıksal operatörler, aritmetik bir sonuç elde etmekte de kullanılırlar. Mantıksal operatörler -1 ve 0 dışında sonuçlar üretebilirler. Doğru/yanlış testlerinde, sıfır yanlış, sıfır dışındaki tüm sonuçlar doğru kabul edilir.

Mantıksal operatörler ikili aritmetikte Boole mantıksal işlemleri için de kullanılabilirler. NOT operatörü kullanılırken işlem sadece tek bir işlenene, sağdakine uygulanır. İşlenenler tam sayı olmalı ve -32768 ile +32767 sayı aralığında yer almalıdır (kesirli sayılar tam sayıya çevrilir). Mantıksal işlemlerin sonucu her zaman tam sayıdır.

Dışlayıcı OR (XOR) operatörü Commodore BASIC'te mevcut değildir: ancak WAIT (BEKLE) yönergesinin bir parçası olarak kullanılır. Dışlayıcı OR (VEYA) eğer iki işlenenin bitleri eşitse 0, değilse 1 sonucunu verir.

İkili aritmetikteki Boole mantıksal işlemleri aşağıda verilen Tablo 1-2'de gösterilmiştir.

İşlenenler, önce 1 ve 0'lardan oluşan 16 bitlik ikili rakamlara çevrilir. Sonra, bir işlenenin her biti ile diğer işlenenin karşılık gelen biti arasında mantıksal işlem yapılarak yeni bir ikili rakam elde edilir.

Tablo 1-2. Boole işlemleri tablosu

AÇIKLAMA	SONUÇ
AND (ve) işlemi her iki bit de 1 ise 1, yoksa 0 sonucunu verir.	1 AND 1 = 1 0 AND 1 = 0 1 AND 0 = 0 0 AND 0 = 0
OR (veya) iki bitten herhangi birisi 1 ise 1, yoksa 0 sonucunu verir.	1 OR 1 = 1 0 OR 1 = 1 1 OR 0 = 1 0 OR 0 = 0
NOT (değil) işlemi bitin mantıksal tamamlayıcısını yani tersini verir.	NOT 1 = 0 NOT 0 = 1
Dışlayıcı OR (XOR) WAIT yönergesinin bir parçasıdır. İki farklı bitin farklı olması halinde 1, aynı olması halinde 0 verir.	1 XOR 1 = 0 1 XOR 0 = 1 0 XOR 1 = 1 0 XOR 0 = 0

AND, OR ve NOT mantıksal operatörleri, operatörün her iki tarafındaki iki işlenen ifadesinde gerçekleştirilecek bir Boole aritmetik işlemi belirtir. NOT koşulunda, sadece sağ tarafta bulunan işlenen göz önüne alınır. İfadede yer alan tüm aritmetik ve bağıntı belirten işlemler tamamlanmadıkça, mantıksal işlemler yapılmaz.

```
10 IF A=100 AND B=100 THEN 20
```

Eğer hem A hem B 100 değerini alırsa sonuç doğrudur.

```
A=96 AND 32 : PRINT A
```

Sonuç A = 32'dir.

```
10 IF A=100 OR B=100 THEN 20
```

Eğer A veya B 100 değerini alırsa sonuç doğrudur.

```
A=64 OR 32 : PRINT A
```

Sonuç A = 96’dır.

```
10 IF NOT X<Y THEN 20
```

Eğer X >= Y sonuç doğrudur.

```
X = NOT 96
```

Sonuç X = -97’dir. İkinin tamamlayıcısı.

İŞLEMLERİN ÖNCELİK SIRASI

İfadelerde belirtilen işlemler belirli bir sıra ile gerçekleştirilirler. Başka bir deyişle bazı işlemler diğerlerine göre öncelik taşırlar. İşlemlerin normal düzenini, iki veya daha fazla işleneni bir parantez içine alarak, yani bir "alt-ifade" yaratarak değiştirebiliriz. Parantez içine alınmış bölümler göz önüne alınmadan önce tek bir değere indirgenmiş olurlar.

Eğer ifadenizde parantez kullanıyorsanız her zaman sağ ve sol parantez sayılarının eşit olmasına dikkat edin. Eğer parantez sayıları eşit değilse ekranda **?SYNTAX ERROR** (yazım hatası) mesajı görüntülenecektir.

Çok düzeyli karmaşık ifadeler oluştururken, parantez içine alınmış bir ifadenin bir bölümü, tekrar tekrar parantez içine alınabilir. Bu, yuvalama (nesting) olarak adlandırılır. İfadelerin yazımında, en çok 10 parantez seti iç içe kullanılabilir. İlk olarak en içte bulunan ifadenin işlemleri yapılır. İfadelere şu örnekleri verebiliriz:

```
A=A+B
```

```
A=C↑(D+E)/2
```

```
A=((X-C↑(D+E)/2)*10)+1
```

```
IF GG$>HH$ THEN PRINT "DOGRU"
```

```
AA$=JJ$+"DAHA"
```

```
K%=1 AND M<>X
```

```
K%=2 OR (A=B AND M<X)
```

```
A=10 NOT (D=E)
```

BASIC yorumlayıcısının normalde izleyeceği işlem sırası şöyledir: Aritmetik işlemler, ilişki belirten işlemler ve son olarak da mantık işlemleri.

Aritmetik ve mantık operatörlerinin kendi içlerinde de bir öncelik sırası vardır. Buna karşılık bağıntı operatörleri, kendi içlerinde bir öncelik sırasına sahip değillerdir ve ifadenin değerlenmesi her zaman soldan sağa doğru yapılır.

Bir ifadede bulunan tüm operatörler aynı öncelik düzeyine sahiplerse işlemler yine soldan sağa doğru yapılır. Parantez içinde yer alan ifadeler üzerinde yapılacak işlemlerde, normal öncelik sırası korunur. Tablo 1-3, aritmetik ve mantık operatörlerinin öncelik sıralarını göstermektedir.

Tablo 1-3. İfadeler üzerinde yapılan işlemlerin öncelik sırası

OPERATÖR	AÇIKLAMA	ÖRNEK
↑	Kuvvet alma	BASE ↑ EXP
-	Olumsuz yapmak (Birli eksi)	-A
*	Çarpma işlemi	AB * CD
/	Bölme işlemi	EF / GH
+	Toplama işlemi	I + 2
-	Çıkarma işlemi	JK - PQ
>=<	Bağıntı operatörleri	A < B
NOT	Değildir mantıksal operatörü	NOT K%
AND	Ve mantıksal operatörü	TX AND 128
OR	Veya mantıksal operatörü	Z OR 10

YAZINSAL DİZİ (STRING) İŞLEMLERİ

Sayısal diziler, sayıları karşılaştırmakta kullanılan (=, < > vb.) bağıntı operatörleri kullanılarak kıyaslanırlar.

Yazınsal diziler karşılaştırılırken, sayıları karşılaştırmada kullanılan operatörler geçerlidir. (=, < >, >=, <=, <, >). Yazınsal dizi karşılaştırmaları, her yazınsal diziden bir karakter alınarak yapılır ve her karakter PET/CBM karakter setindeki koduna çevrilir. Eğer karakter kodları aynıysa karakter de eşit demektir. Karakter kodlarının farklı olduğu koşullarda ise küçük kod numarasına sahip olan karakter, karakter setinde de küçük olarak gösterilir. Karşılaştırma, her iki yazınsal dizideki son karakterlerin de karşılaştırılmasıyla sona erer.

Diğerlerinin eşit olduğu durumlarda, kısa yazınsal dizi, uzun yazınsal diziden daha küçük olarak kabul edilir. Başta ya da arada bırakılan boşluklar da dikkate alınır.

Tüm karşılaştırılmaların sonucunda veri tipi ne olursa olsun, sonuç her zaman bir tam sayıdır. Bu, her iki işlenenin de yazınsal dizi olduğu durumlarda da geçerlidir. Bu özellikten ötürü, yazınsal dizilerin kıyaslamalarından çıkacak sonuç, tek bir işlenen olarak kullanılabilir. Sonuç her zaman -1 ya da 0'dır (Doğru veya Yanlış). Ve sonuç bölen olarak kullanılamaz, çünkü sıfıra bölüm geçerli değildir.

YAZINSAL DİZİ İFADELERİ

İfadeler, onları takip eden bir "<>0" ima edilmiş gibi değerlendirilir. Bu, bir ifade doğruysa, aynı program satırındaki sonraki BASIC deyimlerinin yürütüleceği anlamına gelir. İfade yanlışsa satırın geri kalanı yok sayılır ve programdaki sonraki satır yürütülür.

Sayılarla olduğu gibi yazınsal dizilerle de işlem yapabilirsiniz. CBM BASIC'te geçerli olan tek yazınsal dizi operatörü artı imidir (+). Bu, yazınsal dizilerin yapıştırılmalarında ya da art arda eklenmesinde kullanılır. Bu işlemde artı iminin sağ yanında olan yazınsal dizi, sol taraftakine yapıştırılır ve böylece yeni bir yazınsal dizi elde edilir. Bu sonucu hemen, kıyaslama amacıyla kullanabileceğiniz gibi, bir değişkene de eşitleyebilirsiniz. Eğer bir yazınsal diziyi bir sayısal veriyle kıyaslarsanız **?TYPE MISMATCH** hata mesajını alırsınız. Yazınsal dizi ifadeleri ve yapıştırılmalarına aşağıdaki örnekleri verebiliriz.

```
10 A$="DOSYA": B$="ISMI"
20 C$ = A$ + B$
30 D$ = "YENI "+A$+B$
```

(Satır 20'de “DOSYAISMI” yazınsal dizi oluşur)
(Satır 30'da “YENI DOSYAISMI” yazınsal dizi oluşur)

NOT: YENI sözcüğünün sonundaki (I) harfi ile tırnak imi (") arasındaki boşluğa dikkat edin.

PROGRAMLAMA TEKNİKLERİ
VERİ DÖNÜŞÜMLERİ

CBM BASIC yorumlayıcı gerektiğinde, tam sayı olan bir sayısal veriyi kesirli sayıya veya kesirli sayıyı bir tam sayıya dönüştürebilir. Bu dönüşümün kurallarını şöyle sıralayabiliriz:

- Tüm aritmetik operatörler ve bağıntı operatörleri, tam sayıları ondalık sayıya dönüştürürler. İfadelerin değerlendirilmesinde, tam sayılar önce ondalık sayıya dönüştürülür, ancak sonuç tekrar tamsayı olarak elde edilir. Mantıksal işlemlerde, işlenenleri tam sayıya dönüştürür ve sonuç yine bir tam sayıdır.
- Bir sayısal değişken, başka tip bir sayısal veri tipine eşitlenmişse, sayı değişkenin adına göre, dönüştürülür ve kaydedilir.
- Ondalık bir sayı, tam sayıya dönüştürülürken, noktadan sonra gelen kısım atılır. Sonuç, ondalık sayıdan küçük ya da ona eşittir. Eğer sonuç, +32767 ile -32768 aralığının dışında ise, **?ILLEGAL QUANTITY** (geçersiz sonuç) hata mesajı alırsınız.

INPUT YÖNERGESİNİN KULLANIMI

Artık değişkenlerin ne olduklarını biliyorsunuz. Şimdi bu bilgileri. INPUT (girdi) yönergesinin kullanımına uygulayalım. Böylece, programlama pratiğine de girmiş oluyoruz.

İlk örneğimiz, bir değişkenin saklanmasıyla ilgili. Şimdi N$ değişkenini ele alalım ve bunu bir kişinin ismine eşitleyelim. Artık her PRINT N$ yazışımızda, bilgisayar otomatik olarak bu ismi yazacaktır.

Commodore 64'ünüzde NEW sözcüğünü yazın, **RETURN** tuşuna basın ve şu örneği deneyin.

```
10 PRINT "ISMINIZ" : INPUT N$
20 PRINT "SELAM", N$
```

Bu örnekte, "ISMINIZ" değerini temsil etmek için, N değişkenini kullandık. Koyduğumuz ($) imi ise bilgisayarınıza, bir yazınsal dizi değişkeni kullandığınızı belirtti. Bu noktada, iki değişken türünün tanımlanması çok önemli. Bunlar:

1. Sayısal değişken
2. Yazınsal dizi değişkeni

Sayısal değişkenlerin, 1, 100, 4000 ... gibi sayısal değerleri saklamak için kullanıldığını anımsayacaksınız. Tek bir harf (A), iki harf (AB) veya bir harf ve bir sayıdan (A1) oluşan sayısal değişkenler kullanılabilir. Kısa isimler uzun isimlerden daha az yer kaplar. Örneğin; A, AB'den daha az iç-bellek yitirmenize neden olur. Ayrıca, farklı kategoriler için farklı harf ve sayılar kullanmak pratik bir yarar getirecektir (A1, A2, A3). Sonuçları kesirli sayılardan kurtarmak ve tam sayılar elde etmek isterseniz, kullandığınız değişkenlerin sonuna, bir yüzde işareti koymanız yeterli olacaktır. (AB%, A1% vb.)

Şimdi de, INPUT yönergelerinin kullanımını gösteren birkaç örnek verelim.

```
10 PRINT "BIR SAYI GIR" : INPUT A
20 PRINT A
```

```
10 PRINT "BIR SOZCUK GIR" : INPUT A$
20 PRINT A$
```

```
10 PRINT "BIR SAYI GIR" : INPUT A
20 PRINT A " KERE 5 ESITTIR " A*5
```

NOT: Örnek 3'de, MESAJ veya İSTEKLERİN değişkenlerden farklı, tırnak işareti (" ") içinde olduğu görülmektedir. Değişkenler tırnak içinde olmaz. Ayrıca 20. satırda önce A değişkeninin yazdırıldığına, ardından " KERE 5 EŞİTTİR " mesajının ve ardından hesaplamanın A değişkenini 5 (A*5) ile çarptığına dikkat edin.

Hesaplamalar, birçok programın en önemli kısmını oluştururlar. Hesaplamada, "gerçek sayıları" ya da bunlar yerine değişkenleri kullanmayı seçebilirsiniz. Ancak, kullanıcının verdiği sayılarla çalışıyorsanız, sayısal değişkenler kullanmak zorundasınız. Kullanıcıdan şimdi aşağıdaki gibi iki sayı yazmasını isteyelim.

```
10 PRINT "2 SAYI GIR": INPUT A: INPUT B
```

Şimdi, bu iki sayıyı çarpalım ve C gibi yeni bir sayıyı elde edelim.

```
20 C=A*B
```

Sonucu, mesaj şeklinde şu yönergeyle PRINT ettirebilirsiniz.

```
30 PRINT A "KERE" B "ESITTIR" C
```

Bu üç satırı girin ve programı RUN edin. Mesajların tırnak imi içinde, değişkenlerin ise dışında olmasına dikkat edin.

Şimdi diyelim ki, C değişkeni tarafından gösterilen sayı önünde, bir dolar imi istiyorsunuz ($). $ tırnak imi içinde PRINT edilmelidir. $'ı programınıza eklemek için, **RUN/STOP** ve **RESTORE** tuşlarına basın ve şu satırı yazın:

```
40 PRINT "$" C
```

Şimdi **RETURN** tuşuna basın. RUN yazın ve tekrar **RETURN** tuşuna basın. Dolar imini tırnak içinde yazmalısınız, çünkü C sadece bir sayıyı göstermektedir. C'nin gösterdiği sayı 100 olsun. O zaman, Commodore 64'ünüzün ekranında $100'ü göreceksiniz. Ancak, $C'yi tırnak işareti kullanmadan basmaya (PRINT etmeye) kalkışırsanız, **?SYNTAX ERROR** (Yazım hatası) hata mesajı alırsınız.

Ayrıca, dolar imini temsil eden bir de işken tanımlayarak bunu, istediğiniz sayısal değişkenle kullanabilirsiniz. Örneğin: (önce NEW komutu ile eski programı silin).

```
10 Z$="$"
```

Bu durumda, dolar imini kullanmanız gerektiği anda. Z$ yazınsal dizininin değişkenini kullanabilirsiniz. Şunu deneyin:

```
10 Z$="$" :INPUT A
20 PRINT Z$A
```

Satır 10, Z$ olarak adlandırılan yazınsal dizi değişkenini tanımlar. Burada INPUT'unuz, A dediğiniz sayıdır. Satır 20 ise, Z$ ($), A (sayı)'nın yanına PRINT eder.

Gelir/Gider bütçe örneği

```
5 PRINT "♥"
10 PRINT "AYLIK GELIR": INPUT IN
20 PRINT
30 PRINT "GIDER KATEGORISI 1": INPUT E1$
40 PRINT "GIDER MIKTARI": INPUT E1
50 PRINT
60 PRINT "GIDER KATEGORISI 2": INPUT E2$
70 PRINT "GIDER MIKTARI": INPUT E2
80 PRINT
90 PRINT "GIDER KATEGORISI 3": INPUT E3$
100 PRINT "GIDER MIKTARI": INPUT E3
110 PRINT "♥"
120 E=E1+E2+E3
130 EP=E/IN
140 PRINT "AYLIK GELIR: "IN "TL"
150 PRINT "TOPLAM GIDERLER: "E "TL"
160 PRINT "KALAN MIKTAR: "IN-E "TL"
170 PRINT
180 PRINT E1$"TL TOPLAM HARCAMADAKI %'SI
"(E1/E)*100
190 PRINT E2$"TL TOPLAM HARCAMADAKI %'SI
"(E2/E)*100
200 PRINT E3$"TL TOPLAM HARCAMADAKI %'SI
"(E3/E)*100
210 PRINT
220 PRINT "HARCAMALARIN TOPLAMDAKI %'SI"
EP*100"DIR"
230 FOR X=1 TO 5000: NEXT: PRINT
240 PRINT "TEKRAR YAPALIM MI? (E/H)":INP
UT Y$:IF Y$="E" THEN 5
250 PRINT "♥":END
```

NOT: IN = 0 değildir ve E1, E2, E3 aynı anda 0 olamaz. ♥ simgesini yazmak için **SHIFT** ve **CLR/HOME** tuşlarını kullanın.

Program örneğinin satır satır açıklaması

SATIRLAR	AÇIKLAMA
5	Ekranı temizler.
10	PRINT/INPUT yönergesi.
20	Bir satır boş bırakır.
30	Giderler kategorisi 1 = E1$.
40	Gider miktarı = E1.
50	Bir satır boş bırakır.
60	Giderler kategorisi 2 = E2$.
70	Gider miktarı = E2.
80	Bir satır boş bırakır.
90	Giderler kategorisi 3 = E3$.
100	Gider miktarı = E3.
110	Ekranı temizler.
120	Gider miktarlarını toplar = E.
130	Gider/Gelir yüzdesini hesaplar (%).
140	Geliri görüntüler.
150	Toplam gideri görüntüler.
160	Bakiye görüntüler.
170	Bir satır boş bırakır.
180-200	Her giderin toplam gidere göre %'sini hesaplar
210	Bir satır boş bırakır.
220	Toplam harcamaların %'sini gösterir.
230	Bir süre beklenir.

GET YÖNERGESİNİN KULLANIMI

Birçok basit programda, bilgisayarın veri almak için "INPUT" yönergesini kullanması yeterlidir. Ancak gereksinimlerinizin karmaşıklığı arttıkça, GET yönergesinin kullanımı daha çok önem kazanır. (Örneğin, yazım hatalarını önleyebilirsiniz) GET yönergesi, programlarınıza daha bir esneklik ve "yetenek-zekâ" getirecektir. Bu bölümde, GET yönergesinin kullanımından ve getireceği kolaylıklardan söz edeceğiz.

Commodore 64, 10 karakterlik bir klavye tamponuna (buffer) sahiptir. Bunun anlamı şudur: Bilgisayarınızın, kimi işlemleri yaparken klavyeyi okuması olanaksızdır. Buna karşılık siz, 10 karakterlik bir giriş yapabilirsiniz. Bu karakterler Buffer'da depolanacaktır. Commodore 64'ünüz işi bitince, bunları kullanmaya başlayacaktır.

Bunu görmek isterseniz, Commodore 64'ünüze şu programı yazın:

```
NEW
10 TI$="000000"
20 IF TI$ < "000015" THEN 20
```

Şimdi RUN yazın ve **RETURN** tuşuna basın ve programınız işlerken MERHABA sözcüğünü yazın.

15 saniye kadar değişik bir şey olmayacaktır. Ancak daha sonra MERHABA sözcüğünün ekranda belirdiğini göreceksiniz.

Şimdi, sinemada bilet kuyruğundasınız diyelim. Biletini alıp kuyruktan ilk çıkan insan, kuyruktaki ilk insandır. En son bilet alan ise, sonuncu olandır. GET yönergesinin işleyişi de tıpkı bunun gibidir. Kuyrukta birinci olan karakter, ilk göreceğiniz karakter olacaktır. Eğer gerçekten tuşları kullandıysanız, yazdığınız karakter uygun bir değişkene eşitlenecektir. Eğer kullanmadıysanız, bu değişken boş kalacaktır.

Bu noktada şunu belirtelim: Eğer bir defada 10 karakterden fazlasını depolamak isterseniz, 10 karakterden sonra gelen karakterlerin hepsinin kaybolacağından emin olabilirsiniz.

GET yönergesi, klavye tamponunda bulunan ilk karakteri alır. Klavye tamponu boş ise, yani hiçbir tuşa basılmamışsa, GET yönergesi boş dizi ("") verildiğini kabul eder ve devam eder. Bu nedenle birçok programda, GET yönergesini bir döngü içine yazmanız gerekecektir. Böylece döngü, birisi tuşa basıncaya ve bir karakter yazılıncaya kadar beklemek zorunda kalacaktır.

Aşağıdaki örnek, GET yönergesinin kullanım biçimini göstermektedir. Önceki programı silmek için NEW yazın ve **RETURN** tuşuna basın:

```
NEW
10 GET A$ : IF A$ "" THEN 10
```

("") işaretlerinin arasında aralık olmamasına dikkat edin. Bu, boş bir yazınsal diziyi gösterir ve programı, GET yönergesini tekrarlamak üzere 10'uncu satıra geri gönderir. Bu döngü birisi bir tuşa basıncaya kadar sürecektir.

Yeni bir tuşa basmanızla, programınız satır 10'dan sonra gelen satırla devam edecektir. Programınıza şu satırı ekleyin:

```
100 PRINT A$; : GOTO 10
```

Şimdi programı RUN edin. Ekranda takipçi simgesi (■) görünmeyecektir, ancak yazdığınız karakterler ekranda olacaktır. Bu iki satırlık program, aşağıda gösterilen ekran biçimleme (edit) programının bir parçası olarak kullanılabilir.

Ekran editörü ile birçok iş yapabilirsiniz. Yanıp sönen bir takipçi elde edebilirsiniz. **CLR/HOME** gibi tüm ekranı kazayla silecek tuşları idare edebilirsiniz, hatta isterseniz fonksiyon tuşlarını çeşitli sözcükleri temsil edecek şekilde kullanabilirsiniz. Aşağıdaki program, her fonksiyon tuşunun özel bir amaç için kullanımını sağlar.

```
20 IF A$=CHR$(133) THEN POKE 53280,8 : G
OTO 10
30 IF A$=CHR$(134) THEN POKE 53281,4 : G
OTO 10
40 IF A$=CHR$(135) THEN A$="SAYIN BAY:"+
CHR$(13)
50 IF A$=CHR$(136) THEN A$="SEVGILER,"+C
HR$(13)
```

Parantez içinde yer alan CHR$ sayılarını, EK C'deki CHR$ kod çizelgesinden bulabilirsiniz. Çizelge, her karakter bir sayıya karşılık gelecek biçimde düzenlenmiştir. Dört fonksiyon tuşu, her satırda THEN sözcüğünü izleyen talimatlarca gösterilen işlevi yerine getirmek üzere ayarlanmıştır. Parantez içindeki CHR$ sayılarını değiştirerek, farklı tuşlarla da çalışabilirsiniz. THEN yönergesini izleyen bilgileri değiştirmeniz ise yepyeni talimatların yerine getirilmesini sağlayacaktır.

BASIC PROGRAMLARINI NASIL KISALTABİLİRİZ?

Her programı mümkün olduğunca kısa yapmanız, BASIC programınızın daha yetkin olmasını ve daha çok talimatı içermesini sağlayacaktır. Bu program kısaltma yöntemine "crunching" (kısaltma) adını veriyoruz.

Programın kısaltılması, programlarınız içinde en fazla sayıda komutun yer almasını sağlar. Aynı zamanda, verilen bir boyutta işlemeyen programlarınızın boyutunu küçültmeye yardım eder. Envanter numaraları türünde girişler yapmanız gereken programlarda, kısa programlar bellekte çok az yer kaplayarak, girilecek veriler için daha fazla boş yer bırakacaktır.

Anahtar sözcüklerin kısaltılması:

Anahtar sözcüklerin kısaltmaları EK-A'da verilmiştir. Bu kısaltmalar, satırda girilmesi gereken bilgiyi azaltmak açısından çok yararlıdır. Sık kullanılan kısaltma, soru işareti imidir. Bu im BASIC dilinde PRINT sözcüğüne karşılık gelir. Ancak Commodore 64, kısaltmaların yer aldığı programı listelediğiniz anda, anahtar sözcüklerin açık yazılışlarını görüntüler. Eğer programınızda yer alan bir satır, 80 karakteri (ekranda 2 satır) aşmışsa, anahtar sözcükleri kısaltarak satır değiştirmek isterseniz, önce tüm satırı yeniden girmek zorundasınız. Programın saklanması, bir satırda herhangi bir karakter artışı olmaksızın anahtar sözcüklerin birleştirilmesidir. Çünkü tüm BASIC anahtar sözcükler, Commodore 64 tarafından tanımlanmaktadır.

Satır numaralarının kısaltılması:

Programlara genellikle ilk satıra 100 vererek başlanır ve satır numaraları onar onar artırılır (100, 110, 120). Bu sıralama, program ilerledikçe yeni satırların eklenmesini kolaylaştıracaktır (111, 112, 115). Programları kısaltmanın bir yolu da, satır numaralarının olabildiğince küçük tutulmasıdır (örneğin 1, 2, 3 gibi). Çünkü GOSUB ve GOTO yönergeleri tarafından, gidilmesi belirlenen uzun satır numaraları belekte daha geniş yer kaplarlar. Örneğin 100, bellekte 3 baytlık yer kaplarken, 1 sadece bir baytlık yer kaplar.

Bir satıra birden fazla komut yazabilirsiniz:

Satırı iki nokta ile bölmeniz, o satıra birden fazla komut yazabilmenizi sağlayacaktır. Ancak, bir satırda bulunan karakterlerin toplamı (iki nokta da dahil olmak üzere) 80'i aşmamalıdır. Aşağıda, bir programın kısaltmadan önceki ve sonraki halini gösteren bir örnek verilmiştir.

Kısaltmadan önce:

```
10 PRINT "MERHABA..";
20 FOR T = 1 TO 500: NEXT
30 PRINT "YINE, MERHABA..."
40 GOTO 10
```

Kısaltmadan sonra:

```
10 PRINT"MERHABA..";:FORT=1TO500:NEXT:PR
INT"YINE,MERHABA...":GOTO10
```

Rem yönergelerinin programdan çıkartılması:

REM yönergelerinin yararı, size veya diğer programcılara, programın belirli bir bölümünün işlevini hatırlatmaktır. Programınız tamamlanıp kullanıma hazır hale gelince, REM yönergelerine artık ihtiyacınız kalmayacaktır ve bu yönergeleri çıkartmak bellekte biraz daha yer açılmasını sağlayacaktır. Ancak, aynı program üzerinde ileride tekrar çalışmak niyetindeyseniz, bu programı REM yönergeleriyle birlikte saklamanız yararlı olacaktır.

Değişkenlerin kullanımı:

Eğer bir sözcük, sayı veya tümce programınızda sık sık tekrarlanıyorsa bu uzun sözcükleri ya da sayıları, bir iki harften oluşan bir değişkenle tanımlamanız yararlı olacaktır. Sayılar, tek bir harfle tanımlanabilir. Buna karşılık sözcükler ve tümceler bir harf ve dolar ($) iminden oluşan yazınsal dizi değişkenleri ile tanımlanırlar. Aşağıda buna ilişkin bir örnek veriyoruz.

Kısaltmadan önce:

```
10 POKE 54296,15
20 POKE 54276,33
30 POKE 54273,10
40 POKE 54273,40
50 POKE 54273,70
60 POKE 54296,0
```

Kısaltmadan sonra:

```
10 V=54296:F=54273
20 POKEV,15:POKE54276,33
30 POKEF,10:POKEF,40:POKEF,70
40 POKEV,0
```

Read ve data yönergelerinin kullanımı:

Çok sayıda veri her seferinde, verilerin bir parçası olarak tekrar tekrar yazılabilir, ya da önce ana yol gösterici program yazılır ve kullanılacak veriler DATA yönergesi adı altında uzun bir liste olarak verilir. Bu, özellikle programın uzun sayı listeleriyle dolu olmasını önleyecektir.

Dizi ve matris kullanımı:

Dizi ve matrisler, DATA yönergelerine benzerler. Çok sayıdaki veri bir liste olarak işleme sokulur. Programın verileri kullanan kısmı sırayla bu listeden verileri çeker. Liste çok boyutlu da olabilir.

Boşlukların çıkartılması:

Programınızın boyutlarını küçültmemizin en kolay yollarından birisi de, boşlukların azaltılmasıdır. Gösterdiğimiz örnek programların çoğunda, anlaşılmasını sağlamak amacıyla boşluklar kullandık. Aslında bu boşluklara gerek yoktur, boşlukları azalttıkça yerden kazanırsınız.

GOSUB rutinlerinin kullanımı:

Eğer bir satır ya da komut programınızda sıkça geçiyorsa, bunlar için GOSUB kullanmanız yararlı olacaktır. Böylece bu satırı ya da komutu tekrar tekrar yazmanız gerekmeyecektir.

TAB ve SPC kullanımı:

Bir karakterin ekranda belli bir noktaya yazılması sırasında çeşitli takipçi komutları yazmak yerine, TAB ve SPC talimatlarını kullanmak daha ekonomik olacaktır.

BÖLÜM 2

BASIC DİLİ SÖZLÜĞÜ

- Giriş
- Anahtar-Sözcükleri
- Kısaltmalar ve Fonksiyon Tipleri
- Anahtar-Sözcükleri Açıklaması (Alfabetik Sıraya Göre)
- Commodore 64 Klavye ve Özellikleri
- Ekran Editörü

GİRİŞ

Bu bölümde CBM BASIC dilinin anahtar-sözcükleri anlatılacaktır. İlk önce tüm anahtar-sözcüklerin, kısaltmalarını ve ekrandaki görüntülerini içeren bir liste verilecek, daha sonra ise tüm anahtar-sözcüklerin ayrıntılı açıklaması ve nasır kullanacağınız örneklerle gösterilecektir.

Commodore 64 BASIC'i, anahtar-sözcüklerin birçoğunun kısaltılmasına izin verir. Kısaltmalar, anahtar-sözcüğün ilk bir veya iki harfi ile, izleyen harfin SHIFT ile basılmış şeklinden oluşur.

Yaptığınız kısaltmalar, bellekte fazladan yer kazanmanızı sağlamayacaktır, çünkü, tüm anahtar-sözcükler BASIC yorumlayıcısı tarafından "token" denilen tek bir karaktere indirgenirler. Kısaltmaların da içerildiği programınızı listelediğiniz zaman, tüm anahtar-sözcüklerin ekranda tam yazılışlarının görüntülendiğini fark edeceksiniz. Bir program satırına, eklemek istediğiniz yönergeler için kısaltmaları kullanabilir ve bir satır için geçerli olan 80 karakter sınırını aşabilirsiniz. Ekran editörü, 80 karakterlik satırlar üzerinde çalışır. Bunun anlamı şudur: Eğer satır, kısaltmaları kullandığınız halde 80 satıra erişmişse, listesini aldığınızda, üzerinde birtakım değişiklikler yapma imkânınız yoktur. Değişiklik yapabilmek için ya satırı tüm anahtar-sözcükleri kısaltarak tekrar yazmak, ya da yönergeleri birbirinden ayırıp, farklı satır numaraları vererek yeniden girmektir.

Anahtar-sözcükler, kısaltmalar ve ekrandaki görüntüleri ile birlikte tam bir liste halinde, tablo 2-1'de verilmiştir. Sonraki bölümde ise, Commodore 64'ünüzde kullanılan yönergeler, komutlar ve fonksiyonlar alfabetik sırada, açıklamaları ile birlikte yer alıyor.

Bu bölümde aynı zamanda, BASIC yorumlayıcısı içinde yer alan BASIC fonksiyonları da açıklanmaktadır. "Tanımlı" fonksiyonlar, fonksiyon tanımlanmasına gerek duyulmadan bir programın içinde ya da doğrudan mod yönergelerinde kullanılabilirler. Bunlar "kullanıcı-tanımlı" fonksiyonlardan farklıdır. Tanımlı BASIC fonksiyonları sonuçları, uygun tipteki bir değişkene yerleştirilerek ya da sadece sonuç olarak kullanılabilirler. BASIC 'teki fonksiyon tiplerini ikiye ayırabiliriz:

1. Sayısal
2. Yazınsal dizi

Fonksiyonların argümanları parantez içinde gösterilir. Parantezler fonksiyon anahtar-sözcüğünden hemen sonra gelir. Sol parantez "(" ile anahtar-sözcüğün son harfi arasında boşluk bırakılmamalıdır.

Argüman tipini, son ucun veri tipi belirler. Bir yazınsal diziyle sonuçlanan fonksiyonların son karakterleri, bir dolar ($) işaretidir. Bazı durumlarda yazınsal diziler, bir veya daha fazla sayısal argüman da içerebilirler.

Gerek duyulursa, tam sayılar kesirli sayı, kesirli sayılar ise tam sayı formatına dönüştürülebilir. Tablodan sonra yer alacak açıklamalarda, fonksiyon işleme sokulduğunda alınan sonucun veri tipi verilmiştir. Argümanların veri tipi de, yönergenin formatıyla gösterilmiştir.

BASIC ANAHTAR-SÖZCÜKLER, KISALTMA VE FONKSİYON TİPLERİ

Tablo 2-1. Commodore 64 BASIC anahtar-sözcükleri

KOMUT	KISALTMA	EKRAN	FONKSİYON TİPİ
ABS	A SHIFT + B	A│	SAYISAL
AND	A SHIFT + N	A╱	
ASC	A SHIFT + S	A♥	
ATN	A SHIFT + T	A▌	SAYISAL
CHR$	C SHIFT + H	C▐	YAZINSAL DİZİ
CLOSE	CL SHIFT + O	CL┌	
CLR	C SHIFT + L	C└	
CMD	C SHIFT + M	C╲	
CONT	C SHIFT + O	C┌	
COS	COS	COS	SAYISAL
DATA	D SHIFT + A	D♠	
DEF	D SHIFT + E	D▔	
DIM	D SHIFT + I	D╮	
END	E SHIFT + N	E╱	
EXP	E SHIFT + X	E♣	SAYISAL
FN	FN	FN	
FOR	F SHIFT + O	F┌	
FRE	F SHIFT + R	F▁	SAYISAL
GET	G SHIFT + E	G▔	
GET#	GET#	GET#	
GOSUB	GO SHIFT + S	GO♥	
GOTO	G SHIFT + O	G┌	
IF	IF	IF	
INPUT	INPUT	INPUT	
INPUT#	I SHIFT + N	I╱	
INT	INT	INT	SAYISAL
LEFT$	LE SHIFT + F	LE▁	YAZINSAL DİZİ
LEN	LEN	LEN	SAYISAL

KOMUT	KISALTMA	EKRAN	FONKSİYON TİPİ
LET	L SHIFT + E	L▔	
LIST	L SHIFT + I	L╮	
LOAD	L SHIFT + O	L┌	
LOG	LOG	LOG	SAYISAL
MID$	M SHIFT + I	M╮	YAZINSAL DİZİ
NEW	NEW	NEW	
NEXT	N SHIFT + E	N▔	
NOT	N SHIFT + O	N┌	
ON	ON	ON	
OPEN	O SHIFT + P	O┐	
OR	OR	OR	
PEEK	P SHIFT + E	P▔	SAYISAL
POKE	P SHIFT + O	P┌	
POS	POS	POS	SAYISAL
PRINT	?	?	
PRINT#	P SHIFT + R	P▁	
READ	R SHIFT + E	R▔	
REM	REM	REM	
RESTORE	RE SHIFT + S	RE♥	
RETURN	RE SHIFT + T	RE▏	
RIGHT$	R SHIFT + I	R╮	YAZINSAL DİZİ
RND	R SHIFT + N	R╱	SAYISAL
RUN	R SHIFT + U	R╭	
SAVE	S SHIFT + A	S♠	
SGN	S SHIFT + G	S│	SAYISAL
SIN	S SHIFT + I	S╮	SAYISAL
SPC	S SHIFT + P	S┐	ÖZEL
SQR	S SHIFT + Q	S●	SAYISAL
STATUS	ST	ST	SAYISAL
STEP	ST SHIFT + E	ST▔	

KOMUT	KISALTMA	EKRAN	FONKSİYON TİPİ
STOP	S SHIFT + T	S|	
STR$	ST SHIFT + R	ST_	YAZINSAL
SYS	S SHIFT + Y	S |	
TAB	T SHIFT + A	T♠	ÖZEL
TAN	TAN	TAN	SAYISAL
THEN	T SHIFT + H	T |	
TIME	TI	TI	SAYISAL
TIME$	TI$	TI$	YAZINSAL
TO	TO	TO	
USR	U SHIFT + S	U♥	SAYISAL
VAL	V SHIFT + A	V♠	SAYISAL
VERIFY	V SHIFT + E	V‾	
WAIT	W SHIFT + A	W♠	

Commodore 64 BASIC anahtar-sözcükleri listesi:

ABS	AND	ASC	ATN	CHR$	CLOSE	CLR
CMD	CONT	COS	DATA	DEF	DIM	END
EXP	FN	FOR	FRE	GET	GET#	GOSUB
GOTO	IF	INPUT	INPUT#	INT	LEFT$	LEN
LET	LIST	LOAD	LOG	MID$	NEW	NEXT
NOT	ON	OPEN	OR	PEEK	POKE	POS
PRINT	PRINT#	READ	REM	RESTORE	RETURN	RIGHT$
RND	RUN	SAVE	SGN	SIN	SPC	SQR
STATUS	STEP	STOP	STR$	SYS	TAB	TAN
THEN	TIME	TIME$	TO	USR	VAL	VERIFY
WAIT						

BASIC ANAHTAR-SÖZCÜKLERİNİN AÇIKLAMALARI

ABS (ABSOLUTE: Mutlak değer) (Sayısal fonksiyon)

ABS (<ifade>)

Bir sayının mutlak değerini, yani önünde hiçbir işaret bulunmaksızın sadece değerini verir. Negatif bir sayının mutlak değeri, o sayısının -1 ile çarpılması sonucunda aldığı değerdir.

```
10 X = ABS (Y)
20 PRINT ABS (X * J)
30 IF X = ABS (X) THEN PRINT "POZITIF"
```

AND (AND: Ve) (Operatör)

<ifade> AND <ifade>

İki ayrı işlevi vardır:

1. IF (koşul) yönergelerinde, iki ayrı ifadenin de doğru olması halinde koşulun doğrulanacağını belirtir.

Bir sayının doğru mu yoksa yanlış mı olduğunu saptarken, bilgisayar sayının değeri 0 olmadıkça onu doğru olarak kabul eder. Bir kıyaslama işlemi sırasında, eğer sonuç doğru ise -1 değeri, yanlış ise 0 değeri alır. İkili formatta ise tüm 1'ler için -1, tüm 0'lar için ise 0'dır. Böylece, doğru/yanlış tespitinde kullanılan AND işlemlerinde, doğru olan bitler için sonuç doğru olacaktır.

```
50 IF X=7 AND W=3 THEN GOTO 10: REM YALN
IZ X=7 VE W=3 OLUNCA SONUC DOGRU
60 IF A AND Q=7 THEN GOTO 10: REM YALNIZ
A 0´DAN FARKLI VE Q=7 OLUNCA SONUC DOGRU
```

2. İkili aritmetikte, Boole işlemlerinde kullanılır.

Boole aritmetiğinde eğer AND'lenen her iki sayı da 1 ise AND işleminin sonucu 1'e eşittir. Biri ya da her ikisi de 0 ise sonuç 0'dır, yani yanlıştır.

1 Bitlik AND işlemine örnek

0	1	0	1
AND 0	AND 0	AND 1	AND 1
0	0	0	1

Commodore 64 AND işlemini -32768 ile +32767 arasındaki sayılar için yapabilir. Bu sınırların dışında yer alan sayılar veya kesirli sayılar **?ILLEGAL QUANTITY** hatası ile sonuçlanır. İkili formata dönüştürüldüğünde bu sınırlar, 16 bit ile ifade edilebilecek tüm sayıları içerir. Uygun bitler AND'lendikten sonra, aynı sınırlar içinde 16 bitlik bir sonuç oluştururlar.

16 Bitlik AND işlemine örnek:

```
                      17
                 AND 194
        0000000000010001
    AND 0000000011000010
 (İKİLİ) 0000000000000000
(ONLU)                 0
```

```
                   32007
               AND 28761
        0111110100000111
    AND 0111000001011001
 (İKİLİ) 0111000000000001
(ONLU)             28673
```

```
                    -241
               AND 15359
        1111111100001111
    AND 0011101111111111
 (İKİLİ) 0011101100001111
(ONLU)             15119
```

ASC (ASCII: Karakter değeri) (Sayısal fonksiyon)

ASC (<yazınsal-dizi>)

ASC, yazınsal dizinin ilk karakterinin, 0 ile 255 arasında bir sayı ile gösterilen Commodore ASCII değerini verir. Commodore ASCII değerlerinin tablosu EK C'de verilmiştir.

```
10 PRINT ASC("Z")
20 X = ASC("ZEBRA")
30 J = ASC(J$)
```

Eğer, yazınsal dizinin içinde hiç karakter yoksa **?ILLEGAL QUANTITY** hatası ortaya çıkar. Yukarıdaki üçüncü örnekte J$="" ise, ASC fonksiyonu işlemeyecektir. GET ve GET# yönergeleri CHR$(0)'ı, boş bir yazınsal dizi (null-string) olarak okur. Bu problemi ortadan kaldırmak için, satır 30'da J$ yazınsal dizinin sonuna aşağıda gösterildiği gibi CHR$(0) eklemelisiniz.

```
10 PRINT ASC("Z")
20 X = ASC("ZEBRA")
30 J = ASC(J$ + CHR$(0))
```

ATN (ARC TANGENT: Ark tanjant) (Sayısal fonksiyon)

ATN (<sayı>)

Bu matematiksel fonksiyon, sayının ark tanjant değerini verir. Sonuç, tanjantı verilen sayıya eşit olan, radyan cinsinden açıdır ve daima -π/2 ile + π/2 aralığındadır.

```
10 PRINT ATN (0)
20 X=ATN (J) * 180 / π : REM DERECEYE CE
VIRIR
```

CHR$ (CHARACTER: Karakter) (Yazınsal dizi fonksiyon)

CHR$ (<sayı>)

Bu fonksiyon, bir Commodore ASCII kodunun karakter olarak karşılığını verir. Karakter listesi ve kodları EK C'de verilmiştir. Sayı, 0 ve 255 arasında bir değer olmalıdır. Bu sınırların dışında ise ?ILLEGAL QUANTITY hatası meydana gelir.

```
10 PRINT CHR$(65): REM 65 = BUYUK HARF A
20 A$ = CHR$(13): REM 13 = RETURN TUSU
50 A = ASC(A$) : A$ =CHR$(A): REM C64 AS
CII KODUNA VE GERI DONUSTURULUR
```

CLOSE (CLOSE: Kapat) (G/Ç yönergesi)

CLOSE (<dosya-no>)

Bu yönerge bir veri dosyası ile ilişkiyi kesmek için kullanılır. Kapatılan dosya numarası, OPEN ile açılan dosya numarası ile aynı olmak zorundadır. (OPEN yönergesi için GİRİŞ/ÇIKIŞ Programlama bölümüne bakınız.)

Teyp ve disk sürücü ile çalışılırken CLOSE yönergesi, tampondaki (depodaki) verileri cihaza kaydederek dosyayı kapatır. Eğer bu işlem yapılmazsa, dosya teypte yarım kalır, diskte ise okunamaz duruma gelir. CLOSE işlemi diğer cihazlar için çok fazla gerekli değildir, fakat yeni dosyalar için bellekte yer açılmasını sağlar.

```
10 CLOSE 1
```

```
10 CLOSE X
```

```
10 CLOSE 9*(1+J)
```

CLR (CLEAR: Temizle) (Yönerge)

CLR

Bu yönerge, RAM belleğinin kullanılmış ve daha sonradan gereksinim duyulmayacak yerlerini, kullanıma elverişli duruma getirir. Bellekteki BASIC programına dokunulmadan, tüm değişkenler, GOSUB adresleri, FOR . NEXT döngüleri, kullanıcı tanımlı fonksiyonlar ve dosyalar bellekten silinir ve bu yerler yeni değişkenler için serbest bırakılır.

Diskteki ya da teypteki dosyaların CLR ile kapatılması mümkün değildir. Dosya ile ilgili bilgiler bilgisayarda ya da tamamlanmış tamponlarda kaybolabilir, fakat disk sürücü dosyanın hala açık (OPEN) olduğunu varsayar. Bu konuda daha ayrıntılı bilgi almak için CLOSE deyimine bakınız.

```
10 X = 25
20 CLR
30 PRINT X
RUN
 0

READY.
```

CMD (COMMAND: Komut) (G/Ç yönerge)

CMD <dosya-no> [, yazınsal dizi]

Bu yönerge normalde ekrana gönderilecek olan bilgi çıkışının, dosyada belirtilmiş olan diğer çıkış cihazına gönderilmesini sağlar. Bu dosya; diskette, teypte, yazıcıda ya da modem gibi bir G/Ç cihazında olabilir. Dosya no.'su daha önce OPEN deyiminde kullanılan dosya no.'su ile aynı olmak zorundadır. Yazınsal dizi tanımlandığında dosyaya gönderilir. Bu yöntem, yazıcı çıktılarına başlık koymak için oldukça kullanışlıdır.

CMD komutundan sonra, PRINT ve LIST deyimleri kullanıldığı zaman, metin ekrana değil, aynı formatta dosyanın açılmış olduğu cihaza (Örneğin; yazıcıya) gönderilir.

Çıktının tekrar ekranda yer alması isteniyorsa, CMD ile devreye sokulan dosyaya bir PRINT# yönergesi gönderilmeli ve CLOSE komutu ile dosya kapatılmalı. PRINT#, dosyaya boş bir satır gönderir. Böylece cihaz, veri kabulünü durdurur.

Herhangi bir sistem hatası (**?SYNTAX ERROR** gibi) çıkışın tekrar ekrana dönmesine neden olur. Ancak hata durumundan sonra da dosyaya PRINT# ile boş bir satır göndermeniz gerekir.

```
OPEN 4,4: CMD 4, "BASLIK": LIST: REM PRO
GRAM LISTESI YAZICIYA
```

```
PRINT# 4: CLOSE 4: REM YAZICI LISTE ALMA
Z VE KAPANIR
```

```
10 OPEN 1, 1, 1, "TEST": REM DOSYA ACAR
20 CMD 1: REM CIKISLAR EKRANA DEGIL, TEY
P'E
30 FOR I = 1 TO 100
40 PRINT I: REM TEYP TAMPONUNA SAYILAR Y
ERLESTIRIR
50 NEXT
60 PRINT# 1: REM LISTE ALMAMA KOMUTU
70 CLOSE 1: REM BITMEMIS TAMPONU YAZ, BI
TIR
```

CONT (CONTINUE: Devam) (Komut)

CONT

Bu komut, programların STOP veya END yönergesinden biriyle ya da RUN/STOP tuşuna basılarak durdurulmuş olan işletimlerini tekrar başlatır. Programlar bırakılmış oldukları yerden işlemeye devam ederler.

Program durdurulduğunda, kullanıcı bazı değişkenleri incelemek, değiştirmek veya programa şöyle bir göz atmak isteyebilir. Hataları düzeltirken ya da programı incelerken STOP yönergesi, programın akışının ya da değişkenlerin incelenmesinin sağlanabileceği yerlerde konulmalıdır.

Programda bir değişiklik yaparsanız (hatta takipçi değiştirmediğiniz bir satırın üzerindeyken RETURN'e basarsanız), veya program bir hata yüzünden durmuşsa veya programı tekrar yazmak için CONT yazmadan önce bir hataya neden olduysanız **CAN'T CONTINUE** hatası oluşur ve CONT yönergesi işlemez.

```
10 PI=0: C=1
20 PI=PI+4/C-4/(C+2)
30 PRINT PI
40 C=C+4: GOTO 20
```

Bu program, Pi değerini hesaplar. RUN yazıp bu programı işlettikten kısa bir süre sonra RUN/STOP tuşuna basın. Ekranda:

```
BREAK IN 20
READY.
```

NOT: 20'den farklı bir sayıda olabilir.

Görüntülenecektir. PRINT C yazıp, Commodore 64'ün neler yaptığını görün. Sonra CONT yazıp **RETURN** tuşuna basın Commodore 64'ün kaldığı yerden devam etmesini sağlayın.

COS (COSINUS: Kosinüs) (Fonksiyon)

COS (<sayı>)

Bu matematiksel fonksiyon, radyan cinsinden bir acı olan <sayı>'nın kosinüsünü hesaplar.

```
10 PRINT COS (0)
20 X=COS (Y*π/180):REM DERECEYI RADYANA
CEVIRIR
```

DATA (DATA: Veri) (Yönerge)

DATA <sabit> [, <sabit>, <sabit>, ...]

DATA yönergesi, bir dizi bilginin programın içinde saklanmasını sağlar. Program, bilgileri READ yönergesi ile okur. READ yönergesi, DATA yönergesi ile verilen sabitlerin programın içinde kullanılmasını sağlar.

DATA yönergeleri, READ ile okunmadıkları sürece program tarafından işleme sokulmazlar. Bu yüzden, programın herhangi bir yerine konulabilirler.

Program içindeki tüm data yönergeleri sürekli bir liste gibi işlem görürler. Veriler (data) READ yönergesi ile soldan sağa doğru okunur. Eğer READ yönergesi ile okunmaya çalışılan veri istenilen tipte değilse (örneğin; bir sayı olması gerekirken yazınsal dizi ile karşılaşıldığında) hata oluşur.

Tüm karakterler DATA yönergesi içinde içerilebilirler, fakat bazılarının tırnak ("") içinde yazılması zorunludur. Bunlar: virgül, noktalı virgül gibi noktalama işaretleri, boşluk, (SHIFT)'e basılarak yazılan harfler, grafikler ve takipçi kontrol karakterleridir.

```
10 DATA 1, 10, 5, 8
20 DATA ALI, AYSE, HASAN, ZEYNEP
30 DATA "SAYIN RECEP, NASILSIN, SAYGILAR
, BAY BAY"
40 DATA -1.7E-9, 3.33
```

DEF FN (DEFINED FUNCTION: Tanımlanmış fonksiyon) (Yönerge)

DEF FN <isim> (<değişken>) = <ifade>

Bu yönerge, programda daha sonra da kullanılabilecek bir kullanıcı-tanımlı fonksiyon oluşturmakta kullanılır. Fonksiyon, herhangi bir matematiksel formülden oluşabilir. Kullanıcı-tanımlı fonksiyonlar, programın birçok yerinde kullanılan uzun formüllerin bu şekilde kullanılmasını sağlayarak yerden tasarruf etmenizi sağlar. Formülün bir kez, tanımlama yönergesinde yer alması yeterlidir ve sonra fonksiyon ismi olarak kısaltılır.

Fonksiyon ismi FN harfleri ile ardından gelen bir değişken ismidir. İsim 1 ya da 2 karakterden oluşabilir. İlk karakter harf, ikinci karakter ise sayı ya da harf olabilir.

```
10 DEF FN A(X) = X + 7
20 DEF FN AA (X) = Y * Z
30 DEF FN A9(Q) = INT(RND(1)*Q+1)
```

Fonksiyon daha sonra programın içerisinde, fonksiyon ismi ile parantez içinde bir değişken olarak çağırılır. Bu fonksiyon ismi diğer değişkenler gibi kullanılır ve değeri otomatik olarak hesaplanır.

```
40 PRINT FN A (9)
50 R=FN AA (9)
60 G=G + FN A9 (10)
```

Yukarıdaki 50. satırda parantez içindeki 9 sayısı fonksiyonun sonucunu etkilemez, çünkü 20. satırdaki fonksiyonda parantez içindeki değer kullanılmamıştır. Sonuç Y çarpı Z'dir, diğer iki fonksiyondaki X değerinden bağımsız olarak, parantez içindeki değer sonucu etkiler.

DIM (DIMENSION: Boyut) (Yönerge)

DIM <değişken> (<indisler>) [, <değişken> (<indisler>) …]

Bu yönerge, bir dizi değişkeni (array) tanımlamak ve kaç boyutlu olduğunu ve her bir boyutunun maksimum eleman sayısını belirtmekte kullanılır. Değişken ismi yanına, eğer değişken tamsayı ise %, yazınsal dizi ise $ işareti alır. İndis, dizinin her bir boyutunun en çok kaç eleman alabileceğini gösterir. Tek boyutlu dizilerde bir indis, çok boyutlu dizilerde virgüllerle ayrılmış birkaç indis kullanılır. İndis en fazla 32767 olabilir.

DIM yönergesi her bir dizi için yalnızca bir kez işleme sokulmalıdır. Bir dizi ikinci kez DIM yönergesi ile tanımlandığında, **REDIM'D ARRAY** (önceden tanımlanmış dizi) hatası oluşur. Bu yüzden, çoğu programlarda DIM işlemleri, programın en başında yapılır.

İstediğiniz sayıda DIM yönergesi ve bir dizi içinde 255 indis kullanmanız mümkündür. Bu sayı yalnızca, değişkenleri tutmaya elverişli olan RAM belleğinin kapasitesi ile sınırlandırılmıştır. Dizi yukarıdaki gibi sayısal değişkenlerden veya yazınsal dizilerden veya tam sayılardan oluşabilir. Eğer değişkenler normal kesirli sayılar değilse, değişkenin yazınsal dizi ya da tamsayı değişkeni olduğunu belirtmek için, isimden hemen sonra $ veya % işaretleri kullanın.

Eğer programın içinde, DIM yönergesi ile tanımlanmamış bir dize değişkeni kullanılırsa, otomatik olarak 11 elemanlı bir dize yaratılır.

```
10 DIM A (100)
20 DIM Z (5, 7), Y(3,4,5)
30 DIM Y7% (Q)
40 DIM PH$ (1000)
50 F(4)=9: REM OTOMATIK OLARAK DIM F(10)
TANIMLANIYOR
```

DIM yönergesi kullanarak Futbol skorları tutma örneği:

```
10 DIM S(1,5), T$(1)
20 INPUT "TAKIM ISIMLERI"; T$(0), T$(1)
30 FOR Q = 1 TO 5: FOR T= 0 TO 1
40 PRINT T$(T), "ILK YARI SKORU" Q
50 INPUT S(T,Q): S(T,0)= S(T,0)+S(T,Q)
60 NEXT T,Q
70 PRINT CHR$(147) "SKOR TAHTASI"
80 PRINT "ILK YARI"
90 FOR Q = 1 TO 5
100 PRINT TAB(Q*2+9) Q;
110 NEXT: PRINT TAB(15) "TOPLAM"
120 FOR T = 0 TO 1: PRINT T$(T);
130 FOR Q = 1 TO 5
140 PRINT TAB(Q*2+9) S(T,Q);
150 NEXT: PRINT TAB(15) S(T,0)
160 NEXT
```

DIM'ın kullandığı BELLEĞİN hesaplanması:

	5 bayt dize adı için
	+2 bayt her boyut için
	+2 bayt tam sayıların her elemanı için
VEYA	+5 bayt ondalık sayıların her elemanı için
VEYA	+3 bayt yazınsal dizilerin her elemanı için
VE	+1 bayt yazınsal dizi elemanlarının her bir karakteri için

END (END: Son) (Yönerge)

END

Programın işlemesini durdurur ve kontrolün bilgisayarı kullanan kişiye geçtiğini belirten READY mesajını görüntüler. Bir program içinde, istenilen sayıda END yönergesi yer alabilir. Çok fazla gerekli değilse END yönergesinin sadece programın bitiminde kullanılması önerilir.

END yönergesi STOP yönergesine benzer, tek farkı STOP kullandığınızda ekranda READY yerine **BREAK IN ...** görüntülenmesidir. Her iki yönerge de CONT komutu ile bilgisayarın işleme devam etmesine izin verir.

```
10 PRINT "BU PROGRAMI CALISTIRMAK ISTIYO
R MUSUN?"
20 INPUT A$
30 IF A$ = "HAYIR" THEN END
40 REM PROGRAMIN GERI KALAN KISMI.
999 END
```

EXP (EXPONENT: Üs alma) (Sayısal fonksiyon)

EXP (<sayı>)

Bu matematiksel fonksiyon sabitinin (2.71828183), verilen sayı kadar üssünün çoğalan katını hesaplar. 88.0296919'dan büyük değer **?OVERFLOW** hatasına neden olur.

```
10 PRINT EXP (1)
20 X=Y * EXP (Z * Q)
```

FN (FUNCTION: Fonksiyon) (Sayısal fonksiyon)

FN <isim> (<sayı>)

Bu fonksiyon, önceden bir isimle DEF yönergesi ile tanımlanmış olan formülün kullanımını sağlar. <sayı> formül içindeki yerine (eğer varsa) yerleştirilir ve formül hesaplanır. Sonuç sayısal bir değer olacaktır.

FN fonksiyonu, DEF FN yönergesi işlendikten sonra, direct (doğrudan) modda kullanılabilir.

Eğer FN, onu tanımlayan DEF yönergesinden önce işlenirse **UNDEF'D FUNCTION** (tanımlanmamış fonksiyon) hatası oluşur.

```
PRINT FN A (Q)
```

```
1100 J = FN J (7) + FN J (9)
9990 IF FN B7 (I+1) = 6 THEN END
```

FOR ... TO ... [STEP ...] (FOR: İçin TO: Den STEP: Adım) (Yönerge)

FOR <değişken> = <başla> TO <bitir> [STEP <artma sayısı>]

Bu, değişkenleri sayaç olarak kullanmanızı sağlayan özel bir BASIC yönergesidir. Belirli parametreleri tanımlamanız gerekir: ondalık sayı değişken ismi, başlangıç değeri, bitiş değeri ve her döngüde kaç ekleyeceği gibi.

Aşağıda 1'den 10'a kadar sayan, her sayıyı ekrana basan ve tamamlandığında END ile sona eren ve FOR yönergesi kullanmayan basit bir BASIC programı verilmiştir.

```
100 L = 1
110 PRINT L
120 L = L + 1
130 IF L <= 10 THEN 110
140 END
```

Şimdi de aynı programı FOR yönergesi kullanarak deneyelim.

```
100 FOR L = 1 TO 10
110 PRINT L
120 NEXT L
130 END
```

Gördüğünüz gibi, program daha kısadır ve FOR yönergesi kullanıldığında anlaşılması daha kolay bir hale gelmiştir.

FOR yönergesi işleme sokulduğunda, çeşitli işlemler de yerine getirilir. Önce <değişken>, sayaç olarak kullanılan <başlangıç> değerine eşitlenir. Yukarıda örnekte, L'nin değeri ilk başta 1'dir.

NEXT yönergesine erişildiğinde <artma sayısı>, <değişken>'e eklenir. STEP kullanılmamışsa <artma sayısı>, +1'e eşittir. Yukarıdaki programda 120'nci satıra ilk erişildiğinde, L'ye 1 eklenir, böylece L'nin değeri 2'dir.

Şimdi <değişken> değeri, <bitiş> değeri ile kıyaslanır. Eğer <bitiş> değerinden büyük değilse, program FOR yönergesinden sonra gelen satıra gider. Şimdiki durumda L'nin değeri 2'dir ve <bitiş> değeri olan 10'dan küçüktür ve bu yüzden 110'uncu satıra gidip devam eder.

<değişken>, <bitiş> değerini aştığında, döngü sona erer ve program, NEXT yönergesinden sonra gelen satırdan başlayarak devam eder. Örneğimizde, L'nin değeri 11'e eriştiğinde <bitiş> değeri olan 10'u aşmış olur ve program 130'uncu satırdan başlayarak devam eder.

<artma değeri> pozitif ise <değişken>, <bitiş> değerini aşmalı, negatif ise de <bitiş> değerinden küçük olmalıdır.

NOT: Her döngü en az bir kez işlenir.

```
100 FOR A = 100 TO 0 STEP -1
200 FOR B = PI TO 6 * π STEP .01
300 FOR AA = 3 TO 3
400 NEXT: NEXT: NEXT
```

FRE (FREE: Serbest) (Fonksiyon)

FRE (<değişken>)

Bu fonksiyon, programınız ve değişkenler için RAM'de ne kadar kullanabilir yer kaldığını bildirir. Eğer bir program elverişli olandan daha fazla yeri kullanmaya çalışıyorsa, **?OUT OF MEMORY** (bellek yetersiz) hatasına neden olur. Parantez içindeki sayı herhangi bir değer alabilir ve sonucu etkilemez.

NOT: FRE işleminin sonucu negatif ise, bellekteki elverişli bayt sayısını bulabilmek için FRE sayısına 65536 ekleyin.

```
PRINT FRE (0)
```

```
10 X = (FRE(K)-1000) / 7
950 IF FRE (0) < 100 THEN PRINT "YETERLI
YER YOK"
```

NOT: Aşağıdaki örnek daima o anda elverişli olan RAM'i verir:

```
PRINT FRE(0) - (FRE(0) < 0) * 65536
```

Eğer bu komutu CLR'den sonra kullanırsanız ve elde edeceğiniz sonucu 38909'dan çıkarırsanız, programınızın harcadığı bellek miktarını bulabilirsiniz.

GET (GET: Al) (Yönerge)

GET <değişken> [, <değişken> …]

Bu yönerge, kullanıcının bastığı her tuşu okur. Kullanıcı tuşlara basarken, karakterler Commodore 64'ün klavye deposunda depolanır. Depoda 10 karakterin depolanması mümkündür. 10'uncu karakterden sonra basılan tuş depolanmaz. Karakterlerin bir tanesinin GET yönergesiyle okunması sonucu, diğer karakterler için yer açılır.

Eğer GET yönergesi, sayısal veri için tanımlanmış ise, kullanıcının sayıdan başka bir değer girmesi **?SYNTAX ERROR** hatasına neden olur. Hata olmasını önlemek için, tuşları yazınsal dizi şeklinde okutmak ve bunları sonradan sayıya dönüştürmek daha elverişlidir (BAK. VAL).

GET yönergesi, INPUT yönergesinin getirdiği kısıtlamalara bir alternatif olarak da kullanılabilir. Daha fazla bilgi için programlama teknikleri bölümündeki GET yönergesinin kullanımına bakınız.

```
10 GET A$: IF A$ = "" THEN 10: REM BIR T
USA BASILIN CAYA KADAR BEKLER
20 GET A$, B$, C$, D$, E$: REM 5 TUS OKU
R
30 GET A, A$: REM BIR SAYI VE HERHANGI B
IR TUS
```

GET# (GET#: Dosyadan al) (G/Ç yönerge)

GET <dosya no>, <değişken> [, <değişken> …]

OPEN ile tanımlanmış olan dosyadan her defada bir karakter okur. GET yönergesi ile aynı çalışmasına rağmen veri, klavye haricindeki bir yerden gelir. Eğer hiçbir karakter alınamazsa değişken, boş yazınsal diziye ("") ya da sayısal değerler için 0'a eşitlenir. Dosyalarda verileri birbirinden ayırmak için kullanılan virgül <,> veya **RETURN** tuşunun kodu olan (ASCII kod 13) gibi karakterler, diğer karakterlerle aynı şekilde algılanır.

3 numaralı cihaz (TV ekranı) ile kullanıldığında, bu yönerge karakterleri ekrandan, birer birer okur. GET#'in her kullanılışı takipçiyi 1 hane sağa ilerletir. Mantıksal satırın sonundaki karakter, **RETURN** tuşunun kodu olan CHR$(13)'e dönüştürülür.

```
5 GET# 1, A$
10 OPEN 1, 3: GET# 1, Z7$
20 GET# 1, A, B, C$, D$
```

GOSUB (GO SUBROUTINE: Alt-programa git) (Yönerge)

GOSUB <satır-no>

GOTO yönergesinin özelleştirilmiş bir şeklidir, tek farkla: GOSUB nereden geldiğini bilir. Programın içinde RETURN yönergesiyle (klavyedeki tuşuyla sakın karıştırılmasın) karşılaştığında program, GOSUB yönergesinden hemen sonra gelen yönergeye geri döner.

Alt-programların (subroutine) kullanımına, programın küçük bir kısmının, programın diğer bölümlerinde birkaç kez kullanılması durumunda gereksinim duyulur. Alt-programları kullanarak aynı satırların değişik yerlerde gereksiz yere tekrar tekrar yazılması önlenmiş olur. Böylelikle, programlarınız için size ayrılmış olan yerden de tasarruf etmiş olursunuz. Bu yüzden GOSUB, DEF FN yönergesine benzer. DEF FN, bir formülü kullanırken, GOSUB'da çok satırlı rutinleri kullanırken yer kazandırır.

GOSUB'un kullanılmadığı örnek:

```
100 PRINT "BU PROGRAM"
110 FOR L = 1 TO 500 : NEXT
120 PRINT "BIR DONGU KULLANARAK"
130 FOR L = 1 TO 500 : NEXT
140 PRINT "EKRANA YAVAS YAVAS"
150 FOR L = 1 TO 500 : NEXT
160 PRINT "BIRSEYLER YAZAR:"
170 FOR L = 1 TO 500 : NEXT
```

GOSUB'un kullanıldığı örnek:

```
100 PRINT "BU PROGRAM"
110 GOSUB 200
120 PRINT "BIR DONGU KULLANARAK"
130 GOSUB 200
140 PRINT "EKRANA YAVAS YAVAS"
150 GOSUB 200
160 PRINT "BIRSEYLER YAZAR:"
170 GOSUB 200
180 END
200 FOR L = 1 TO 500: NEXT
210 RETURN
```

Program GOSUB deyimine eriştiği her an, satır numarası ve program içindeki pozisyonu, belleğin 256 baytını kaplayan "yığın" (stack) diye tanımlanan alanda saklanır. Bu, yığın içinde saklanacak olan veri sayısını kısıtladığından, saklanacak olan alt program dönüş adreslerinin sayısı sınırlıdır. Bu yüzden, her GOSUB yönergesine karşılık gelen bir RETURN yönergesinin bulunması gerekir, yoksa bellekte daha bir sürü boş yeriniz olduğu halde bellekten taşmış olacaksınız.

GOTO (GO TO: Git) (Yönerge)

GOTO <satır numarası> veya GO TO <satır numarası>

Bu yönerge, BASIC programın satır numarası sırası dışında çalışabilmesini sağlar. GOTO deyimi ve yanındaki sayı, programın bu sayı ile başlayan satıra atlamasına neden olur. GOTO deyiminden sonra bir sayı yoksa bu GOTO 0 olarak kabul edilir.

GOTO ile hiç bitmeyecek döngüler oluşturmak mümkündür. En basit örnek: 10 GOTO 10 gibi sürekli kendine giden satırdır. Bu tür döngüler, klavyedeki **RUN/STOP** tuşu ile durdurulabilir.

```
GOTO 100
```

```
10 GO TO 100
```

```
10 GOTO 100
```

IF … THEN … (IF: Eğer THEN: Sonra) (Yönerge)

IF <deyim> THEN <satır-no>
IF <deyim> GOTO <satır-no>
IF <deyim> THEN <yönerge>

Bu yönerge, BASIC'e "zekasını" yani durumları değerlendirme yetisini ve sonuca göre yapacağı değişik hareketleri seçmesini sağlayan yönergedir.

IF kelimesinin ardından gelen deyim, değişkenleri, yazınsal dizileri, sayıları, kıyaslamaları ve mantıksal operatörleri içerebilir. THEN kelimesi, aynı satırda yer alır ve bu kelimeden sonra da bir satır numarası, bir ya da birden fazla BASIC yönergesi gelir. Eğer deyimin sonucu yanlış ise, THEN kelimesinden sonra gelen işlemler yapılmaz ve program alttaki satıra atlar. Sonuç doğru ise, programın THEN kelimesinden sonra gelen satıra dallanmasını ya da bu satırda yer alan diğer yönergelerin işlenmesini sağlar.

```
100 INPUT "BIR SAYI GIRIN"; N
110 IF N <= 0 GOTO 200
120 PRINT "KAREKOKU=" SQR(N)
130 GOTO 100
200 PRINT "SAYI 0'DAN BUYUK OLMALI"
210 GOTO 100
```

Bu program pozitif bir sayının karekökünü hesaplar. Burada IF deyimi, girilen sayıyı kontrol etmek için kullanılır. Eğer N<=0 doğru ise, program 200'ncü satıra dallanır. Yanlış ise bir sonraki satır, yani 120'nci satır işleme sokulur. Dikkat ettiyseniz IF ... THEN yönergesinde THEN GOTO kullanımına gerek yoktur. 110'uncu satırda olduğu gibi GOTO 200, THEN GOTO 200 anlamına gelmektedir.

```
100 FOR L = 1 TO 100
110 IF RND(1)< .5 THEN X=X + 1 :GOTO 130
120 Y = Y + 1
130 NEXT L
140 PRINT "YAZI= " X
150 PRINT "TURA= " Y
```

110'uncu satırdaki IF rastgele sayının .5'den küçük olup olmadığını kontrol eder. Eğer sonuç doğru ise, THEN kelimesinden sonra gelen tüm işlemler yerine getirilir. Önce X bir arttırılır, sonra program 130'uncu satıra dallanır. Eğer yanlış ise, program bir sonraki satırı, yani satır 120'yi işleme sokar.

INPUT (INPUT: Giriş) (Yönerge)

INPUT ["<uyarı metni>";] <değişken> [, <değişken>, …]

Bu, programla uğraşan kişinin, bilgisayara bilgi girmesini sağlayan bir yönergedir. İşleme sokulduğunda bu yönerge, ekrana bir soru işareti (?) basar ve takipçi soru işaretinin bir sağında yer alır. Şimdi bilgisayar, takipçi yanıp sönerken, bilgi yazılıp **RETURN** tuşuna basılıncaya kadar beklemektedir.

INPUT sözcüğünün ardından tırnak işaretleri ("") arasında yazı yazmak mümkündür. Bu yazı ekranda, soru işaretinden önce yer alır.

Yazı yani (uyarı)'dan sonra noktalı virgül (;), daha sonra da bir ya da aralarına virgül koyarak birden fazla değişken ismi yer alır. Kullanıcının girdiği sayı veya yazı, bu değişkene atanır. Değişken, kurallara uygun olmak şartıyla herhangi bir değişken ismi olabilir. Farklı değişken isimleri kullanarak, birden fazla bilgi girişi yapmanız da mümkündür.

```
100 INPUT A
110 INPUT B, C, D
120 INPUT "BILGI GIR"; E
```

Bu program RUN yazılarak işletildiğinde, 100'üncü satır için Commodore 64'ünüzün bilgi beklediğini göstermek için ekranda soru işareti belirir. Girdiğiniz sayı daha sonra program içinde kullanılmak üzere A değişkenine yerleştirilir. Eğer girdiğiniz bilgi sayı değilse **?REDO FROM START** (baştan yapınız) hata uyarısı ekranda görüntülenir. Bu, bilgisayarın sayı beklerken yazınsal dizi girilmesi nedeniyle ortaya çıkmıştır. Eğer hiç bilgi girilmeden RETURN tuşuna basılırsa değişkenin değeri değişmez. Yani daha önce A = 5 ise, A gene 5 olarak kalır.

Şimdi 110'uncu satır için diğer bir soru işareti ekranda görüntülenecektir. Bir sayı yazıp RETURN 'e bastığımızda ekranda, daha fazla bilgi girilmesini beklediğini belirten iki soru işareti (??) görüntülenir. İki soru işaretinin görüntülenmesini, gerekli sayıdaki bilgiyi, aralarına virgül koyarak girdiğinizde önleyebilirsiniz. Fakat INPUT yönergesinin gereksindiği sayıdan fazla giriş yaparsanız bu. **?EXTRA IGNORED** (ekstralar iptal edildi) hatasına neden olacaktır. Bu hata, herhangi bir değişkenin içinde yerleştirilemeyecek bilgiler girdiğinizde ortaya çıkar.

Satır 120'de ise soru işaretinden önce "BILGI GIR" uyarısı görüntülenecektir. Uyarı ile değişken listesi arasına noktalı virgül (;) koymak gerektiğini sakın unutmayın.

INPUT yönergesi program dışında, yani doğrudan modda asla kullanılamaz. Commodore 64'ün INPUT değişkenleri tamponu (buffer) için alana gereksinimi vardır ve aynı alan komutlar için de kullanılmaktadır.

INPUT# (INPUT#: Dosyadan giriş) (G/Ç yönerge)

INPUT# [“<dosya numarası>”;] <değişken> [, <değişken>, …]

Bu, disket ya da kaset üzerindeki bir dosyadan bilgi almak için kullanılan en hızlı ve en kolay yoldur. Veri 80 karakterlik uzunluktaki değişkenler şeklindedir. Bu, GET#, yönergesindeki her seferde bir karakter okunması yönteminin karşıtıdır. Dosyanın önceden OPEN deyimi ile açılmış olması gerekir. INPUT# yönergesinin değişkenlere değer yüklemesi daha sonraki işlemdir.

INPUT# komutu, RETURN kodu (CHR$(13)), bir virgül (,), noktalı virgül (;) veya iki nokta üst üste (:) okuduğunda değişkenin bittiğini varsayar. Bu karakterlerin kullanımına gerek duyulduğunda ise, tırnak işaretlerini kullanabilirsiniz. (PRINT# yönergesine bakınız).

Eğer kullanılan değişken tipi sayısal ise, sayısal olmayan bir karakter algılandığında **BAD DATA** hatası oluşur. INPUT#, 80 karakter uzunluğundaki dizileri okuyabilir. Daha uzun bir dizi okunmak istenildiğinde de **STRING TOO LONG** (Dizi çok uzun) hatası oluşur.

3 no.’lu cihaz (ekran) ile birlikte kullanıldığında, bu yönerge tüm satırı okuyacak ve takipçiyi aşağıya bir sonra gelen satırın üzerine ilerletecektir.

```
10 INPUT# 1, A
20 INPUT# 2, A$, B$
```

INT (INTEGER: Tam sayı) (Sayısal fonksiyon)

INT (<sayısal>)

İfadenin tam sayı değerini verir. Eğer ifade pozitif ise, noktadan sonraki kesirli kısmı hesaba katılmaz. Eğer negatif ise de kesirli kısım, sonucun bir önceki tam sayıya eşit olmasına sebep olur.

```
120 PRINT INT(99.4343),INT(-12.34)
RUN
 99        -13

READY.
```

LEFT$ (LEFT: Sol) (Yazınsal dizi fonksiyon)

LEFT$ (<yazınsal dizi>,<sayısal>)

Yazınsal dizinin soldan itibaren, <sayısal> kadar olan kısmını yazınsal dizi olarak verir. Tam sayı 0 ile 255 arasında bir değer olmalıdır. Eğer sayı dizi içindeki toplam karakter sayısından büyük ise, cevap yazınsal dizinin kendisidir. Eğer tam sayı sıfır ise de sonuç sıfır uzunluktaki boş dizidir (null string).

```
10 A$ = "COMMODORE BILGISAYARLARI"
20 B$ = LEFT$(A$, 9): PRINT B$
RUN
COMMODORE

READY.
```

LEN (LENGTH: Uzunluk) (Sayısal fonksiyon)

LEN (<yazınsal dizi>)

Basılmamış karakterler ve boşluklar dahil, yazınsal dizi içindeki karakterlerin sayısını verir.

```
CC$= "COMMODORE BILGISAYAR": PRINT LEN(C
C$)
RUN
 20

READY.
```

LET (LET: Ver) (Yönerge)

[LET] <değişken> = <ifade>

LET yönergesi, bir <değişken>'e değer atamak için kullanılır. Fakat LET kelimesini kullanmak kullanıcının isteğine bağlı olduğundan, uzman kullanıcılar bellek kaybına neden olmamak için LET kelimesini kullanmazlar. Bir değişkene değer atamak için yalnızca eşit işareti (=) yeterlidir.

```
10 LET D = 12
20 LET E$ = "ABC"
30 F$ = "SOZCUKLERI"
40 OPLAM$= E$ + F$
```

(Satır 10'a, 10 D=12 yazsak her ikisi aynıdır).
(Satır 40'daki OPLAM$, ABCSOZCUKLERI'ne eşittir).

LIST (LIST: Liste) (Komut)

LIST [[<ilk satır no>] - [<son satır no>]]

LIST komutu size, o anda Commodore 64'ün belleğinde bulunan BASIC programının satırlarını görüntüleme olanağı verir. Bu, çabuk ve kolayca listelediğiniz programları incelemek ve değiştirmek için bilgisayarınızın güçlü ekran editörünü kullanmanızı sağlar.

LIST sistem komutu, o anda kullanılan çıkış cihazında, bilgisayarın belleğinde bulunan programın parçalarının ya da tümünün görüntülenmesini sağlar. LIST, normalde ekrana yöneliktir ve CMD yönergesi ile disket ya da yazıcıya yönlendirilebilir. Her LIST komutu uygulandığında, bilgisayar direkt moda geçerek READY verecektir.

LIST sırasında ekrandaki geçişi yavaşlatmak için **CTRL** tuşuna basabilirsiniz, **RUN/STOP** tuşu ise listeyi durduracaktır.

Herhangi bir satır numarası verilmezse, programın tamamı listelenir. Yalnızca ilk satır numarası belirtilirse ve ardından bir tire (-) gelirse, o satır ve tüm yüksek numaralı satırlar listelenir. Yalnızca son satır numarası belirtilmişse ve önünde bir tire varsa, programın başından bu satıra kadar olan tüm satırlar listelenir. Her iki numara da belirtilirse, Listelenen satır numaraları dahil tüm aralık görüntülenir.

```
LIST
```

(Bellekteki tüm programı listeler).

```
LIST 500
```

(Sadece satır 500'ü listeler).

```
LIST 150-
```

(150'inci satır ve sonrasının tümünü listeler).

```
LIST -1000
```

(1000'inci satır ve öncesinin tümünü listeler).

```
LIST 150-
```

(150 ile 1000'inci satır ve arasının tümünü listeler).

```
10 PRINT "BU 10'UNCU SATIR"
20 LIST : REM PROGRAM MODUNDA LIST KULLA
NILABILIR
30 PRINT "BU 30'UNCU SATIR"
```

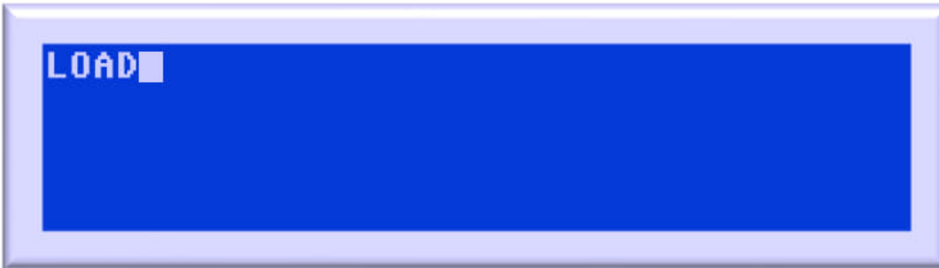
LOAD (LOAD: Yükle) (Komut)

LOAD ["<dosya-adı>"] [,<aygıt>] [,<adres>]

Bu yönerge kaset ya da disket üzerindeki program dosyasının içeriğini, bilgisayarın belleğine yükler. Bu yolla, yüklenmiş olan bilgiyi kullanabilir ya da değiştirebilirsiniz. Cihaz numarası vermek isteğe bağlıdır. Cihaz numarası belirtmezseniz bilgisayar onu 1 olarak kabul eder, bu da teybin cihaz numarasıdır. Disk sürücünün cihaz numarası 8'dir. LOAD, açık olan tüm dosyaları kapatır ve eğer doğrudan (direct) modda kullanılıyorsa, programı okumadan önce CLR (temizlik) işlemi yapar. Eğer LOAD programın içinden yapılıyorsa, program işletilir. LOAD komutu ile birden fazla programı arka arkaya çalıştırabilirsiniz. Bu sırada bellekte sadece o anda çalışan program bulunur. Zincirleme operasyonu ("chain" operation) denilen bu işlem sırasında hiçbir değişken sıfırlanmaz, yani eski değerlerini korur.

Eğer işlemi, (disket kullanırken) dosya adının kısmi uyumunu (file-name partern matching) kullanarak yapıyorsanız, bir parçası verilmiş dosya adına uyan ilk program yüklenir. Tırnak içindeki yıldız ("*") işareti, disket listesinde yer alan programlardan ilkinin yüklenmesini sağlar. Eğer verilen isim diskette yer almıyorsa veya program dosyası değilse bu, **?FILE NOT FOUND** (dosya bulunamadı) hatasına neden olur.

Programları teypten yüklerken (dosya adı) verilmesi zorunlu değildir. Bu durumda kaset üzerinde o anda hangi program varsa o programı yükleyecektir. Commodore 64, PLAY tuşuna basıldıktan sonra ekranı temizleyecek ve rengini çerçeve rengine dönüştürecektir. Program bulunduğunda ekranın rengi zemin rengine dönüşür ve "FOUND" (bulundu) mesajı ekranda görüntülenir. C= tuşu, CTRL tuşu, ← tuşu veya ARALAMA tuşuna basıldığında dosya yüklenir. İkincil (adres) 1 olarak verilmediği sürece programlar belleğin 2048'inci yerleşiminden başlayarak yüklenecektir. Eğer ikincil adres olarak 1 verilmişse program belleğin hangi yerleşiminde iken saklanmışsa o yerleşime yüklenir.



(Kasette yer alan programı yükler).



(A$ içinde yer alan ismi kullanarak programı yükler).

```
LOAD "*",8
```

(Disketteki ilk programı yükler).

```
LOAD "",1,1
```

(Kasetteki ilk programa bakar ve nereden geldiyse belleğin aynı yerine yükler).

```
LOAD "YILDIZ SAVASI"

PRESS PLAY ON TAPE

FOUND YILDIZ SAVASI
LOADING

READY.
```

(Kasetten YILDIZ SAVASI isimli programı yükler).

```
LOAD "YILDIZ SAVASI",8

SEARCHING FOR YILDIZ SAVASI
LOADING


READY.
```

(Disketten YILDIZ SAVASI isimli programı yükler).

```
LOAD "YILDIZ SAVASI",8,1

SEARCHING FOR YILDIZ SAVASI
LOADING


READY.
```

(Disketten YILDIZ SAVASI isimli programı saklanmış olduğu belirli bellek yerleşimine yükler).

```
LOAD "??M????R*",8,1
```

(Disketten 3'üncu harfi M, 8'inci harfi R olan, belirsiz uzunluktaki ilk programı yükleyecektir. Bu örneğin, COMMODORE ya da COMPUTER isimli program olabilir.)

LOG (LOGARITHM: Logaritma) (Sayısal fonksiyon)

LOG (<sayısal>)

<sayısal> değerin, e tabanına göre doğal logaritmasını verir. Eğer <sayısal> değer 0 veya negatif ise bu, **?ILLEGAL OUANTITY** hatasına neden olur.

```
25 PRINT LOG(45/7)
RUN
 1.86075234

READY.
```

```
10 NUM=LOG(ARG)/LOG(10)
```

(ARG'nin 10 tabanına göre logaritmasını verir.)

MID$ (MIDDLE: Orta) (Yazınsal dizi fonksiyon)

MID$ (<yazısal dizi>, <sayısal-1> [,<sayısal-2>])

MID$ fonksiyonu, daha büyük bir <yazınsal dizi> değişkeninin bir kısmını alt-dizi şeklinde verir. Bu alt-dizinin başlama noktası, <sayısal-1>'de verilen değer ile, uzunluğu ise <sayısal-2>'de verilen değer ile tanımlanır. Sayısal değişkenlerin her ikisi de 255'ten büyük, 0'dan küçük olamaz.

Eğer <sayısal-1>'in değeri <yazınsal dizi>'nin uzunluğundan büyük ise ya da <sayısal-2>'nin değeri 0 ise, MID$ fonksiyonun değeri boş diziye (null string) eşittir. <sayısal-2> olarak bir değer verilmemişse, bilgisayar bunu dizinin uzunluğu olarak kabul eder. Ve eğer, <yazınsal dizi>'nin başlangıç pozisyonundan sonuna kadar olan uzunluğu, <sayısal-2>'de verilen değerden küçük ise, dizinin geri kalan kısmı kullanılır.

```
10 A$="IYI"
20 B$="SABAHLAR AKSAMLAR"
30 PRINTA$ + MID$(B$, 9, 9)
RUN
IYI AKSAMLAR

READY.
```

NEW (NEW: Yeni) (Komut)

NEW

NEW komutu o anda bilgisayarın belleğinde yer alan programı bellekten silmek ve programın içinde kullanılmış olan tüm değişkenleri sıfırlamak için kullanılır. Yeni bir programı yazmaya başlamadan önce belleği temizlemek için doğrudan modda NEW komutunun girilmesi gerekir. NEW komutunu program içinde de kullanılabilir fakat, daha önce yaptığınız her şeyin bellekten silineceğini unutmayınız. Bu, özellikle programdaki hataları düzeltmeye çalıştığınızda zorluk çıkacaktır.

DİKKAT: Eski programınızı bellekten silmeden yeni program yazdığınızda, bu iki programın karışmasına neden olacaktır

```
NEW
```

(Programı ve değişkenleri siler.)

```
10 NEW
```

(Programı durdurur ve bellekten programı ve değişkenleri siler.)

NEXT (NEXT: Sonra) (Yönerge)

NEXT [<sayaç>] [,<sayaç>] …

NEXT yönergesi, FOR ... NEXT döngüsünün bitimini sağlamak için FOR ile birlikte kullanılır. NEXT'in fiziksel olarak döngüdeki son deyim olması gerekmez, ancak her zaman bir döngüde yürütülen son deyimdir. <sayaç>, döngüyü başlatmak için FOR ile kullanılan döngü dizininin değişken adıdır. Tek bir NEXT, her bir FOR'nun <sayaç> değişken adı/adları tarafından takip edildiğinde, birkaç iç içe döngüyü durdurabilir. Bunu yapmak için her bir ad, en içteki döngüden en sondaki döngüye doğru sırada görünmelidir. Birkaç değişken adını artırmak ve durdurmak için tek bir NEXT kullanırken, her değişken adı virgülle ayrılmalıdır. Döngüler 9 seviyeye yuvalanabilir. Sayaç değişkeni/değişkenleri atlanırsa, geçerli düzeyin (iç içe döngülerin) FOR'uyla ilişkili sayaç artırılır.

NEXT'e ulaşıldığında, sayaç değeri 1 veya isteğe bağlı bir STEP değeri ile artırılır. Daha sonra döngüyü durdurma zamanının gelip gelmediğini görmek için bir son değere karşı test edilir. Sayaç değeri, bitiş değerinden büyük olan bir NEXT bulunduğunda, bir döngü durdurulacaktır.

```
10 FOR J=1 TO 5: FOR K=10 TO 20: FOR N=5
TO-5 STEP-1
20 NEXT N, K, J
```

(İç içe döngüleri durdurma.)

```
10 FOR L=1 TO 100
20 FOR M=1 TO 10
30 NEXT M
40 NEXT L
```

(Döngülerin birbirini nasıl geçmediğini dikkat edin.)

```
10 FOR A=1 TO 10
20 FOR B=1 TO 20
30 NEXT
40 NEXT
```

(Değişken isimlerine gerek olmadığına dikkat edin.)

NOT (NOT: Değil) (Mantıksal operatör)

NOT <ifade>

Belirtilen <ifade>'nin ikili sistemdeki gösteriminde oluşan bitleri ters çevirerek oluşan sayıyı verir. İfadenin sonucu önce tam sayıya çevrilir. -32768 ile +32767 arasında olmalıdır. IF ... THEN ... yönergelerinde de kullanılır.

Mantıksal NOT operatörü, tek işlenenindeki her bitin değerini "tamamlar" ve bir tamsayı "ikinin tümleyeni" sonucu üretir. Başka bir deyişle, NOT gerçekten "eğer değilse..." demektedir. Kesirli bir sayıyla çalışırken, işlenenler tam sayılara dönüştürülür ve kesirler kaybolur. NOT operatörü, bir mantıksal işlemin sonucu olan doğru/yanlış değerini tersine çevirmek için bir karşılaştırmada da kullanılabilir ve bu nedenle karşılaştırmanın anlamını tersine çevirir. Aşağıdaki ilk örnekte, "AA"'nın "ikiye tümleyeni" "BB"'ye eşitse ve "BB", "CC"'ye eşit DEĞİLSE, ifade doğrudur.

```
10 IF NOT AA = BB AND NOT (BB=CC) THEN E
ND
```

```
NN%=NOT 96: PRINT NN%
-97

READY.
```

NOT: NOT değerini bulmak için X=(-(X+1)) ifadesini kullanın. (Herhangi bir tam sayının ikisinin tümleyeni, bit tümleyeni artı birdir.)

ON (ON: Üzerinden) (Yönerge)

ON <değişken> GOTO/GOSUB <satır no> [,<satır no>] ...

ON yönergesi, <değişken>'in değerine bağlı olarak satır numaralarından bir tanesine dallandırmasını sağlamak amacıyla kullanılır. Değişkenin değeri, 0 ile verilmiş olan satırların sayısı arasında yer alır. Eğer değer tamsayı değilse, kesirli kısım hesaba katılmaz. Örneğin, değişkenin değeri 3 ise, ON yönergesi, listesindeki üçüncü satır numarasına gidilmesini sağlayacaktır.

Değişkenin değeri negatif ise bu, **?ILLEGAL OUANTITY** hatasına neden olur. Eğer sayı 0 ise ya da listede yer alan satır nolarının sayısından büyük ise. Bu yönerge işlemez ve program bir sonraki yönergeden itibaren devam eder.

ON gerçekte, IF ... THEN ... yönergesinin değişik bir kullanım şeklidir. Programı belirli bir satıra gönderebilmek için bir dolu IF yönergesi kullanmak yerine, bir ON yönergesi bir liste halindeki IF yönergeleri ile yer değiştirebilir. İlk örnekte görüldüğü gibi bir ON yönergesi, 4 tane IF ... THEN ... yönergesinin yerine kullanılabilir.

```
ON -(A=7)-2*(A=3)-3*(A<3)-4*(A>7) GOTO 5
0,60,70,40
```

```
ON X GOTO 100,130,180,220
```

```
ON X GOSUB 9000,20,9000
```

```
100 ON NUM GOTO 150, 300, 320, 390
500 ON TPL / 2 + 1 GOSUB 50, 80, 20
```

OPEN (OPEN: Aç) (G/Ç yönergesi)

OPEN <dosya-no>, [<cihaz>] [,<adres>] [,"<dosya ismi> [,<tip>] [,<mod>]"]

Bu yönerge bir dış cihaza, giriş ve/veya çıkış için kanal açmakta kullanılır. Fakat, her OPEN yönergesi için yukarıdaki formatın tümünü kullanmak gerekmez. Bazı OPEN yönergelerinde aşağıda verilmiş olan 2 kod yeterli olacaktır.

1. Dosya numarası
2. Cihaz numarası

<dosya-no>, OPEN, CLOSE, CMD, GET#, INPUT# ve PRINT# yönergelerinin her birinde kullanılan bir dosya numarasıdır ve kullanılan cihaz ve dosya isimlerini birleştirir. Dosya no 1 ile 255 arasında istediğiniz bir sayı olabilir.

NOT: 128'in üzerindeki sayılar başka kullanımlar için tasarlandığından, dosya numarası için 127'nin altındaki sayıları kullanmanız önerilir.

Sistemdeki her cihazın <yazıcı, disk sürücü, teyp) bir numarası vardır ve <cihaz-no> OPEN yönergesinde veri dosyasının yer aldığı cihazı tanımlamak için kullanılır. Teyp, disk sürücü ya da yazıcılar ikincil adrese gerek duyabilirler. Bunları cihaza, ne yapılması gerektiğini bildiren kodlar olarak düşünebiliriz. Cihazın dosya numarası GET#, INPUT# ve PRINT# yönergelerinin her birinde kullanılır.

Eğer <cihaz-no> verilmemişse, bilgisayar bilginin otomatik olarak 1 nolu cihaz olan teybe gönderileceğini ve teypten alınacağını varsayar. Dosya isminin de verilmesi zorunlu değildir. Fakat daha sonra programın içinde, isim vermemiş olduğunuz dosyayı isim ile çağırmanız mümkün değildir. Dosyaları kasete sakladığınızda eğer ikincil adres vermediyseniz, bilgisayar bu adresin sıfır (0) olduğunu varsayar.

İkincil adresin değerinin bir (1) olması kasetteki dosyaların yazmak için açılmasını sağlar. İki (2) değeri, dosya daha sonra kapatıldığında teyp-sonu işaretinin konulmasına neden olur. Teyp-sonu işareti, **?DEVICE NOT PRESENT** hatasına neden olan, son veriden sonrasının istenmeden okunmasını önler.

Disketteki dosyalarda, 2 ile 14 arasındaki ikincil adresler veri dosyaları için elverişlidir. Bu sınırın dışındaki sayıların ise DOS komutlarında özel anlamları vardır. Disk sürücünüzü kullanırken bir ikincil adres kullanmak zorundasınız. (D0S komutları konusunda daha detaylı bilgi edinmek istiyorsanız disk sürücü el kitabına bakınız.)

<dosya-ismi> 1 ila 16 karakter uzunluğunda bir yazınsal dizidir ve kaset ve yazıcı için kullanmak sizin istediğinize bırakılmıştır. Dosya <tip>'i tanımlanmamış ise, <mod> verilmedikçe bunun program dosyası olduğu varsayılır. Sequential <sıralı> dosyalar, modu, yazmak için <mod> = W olarak tanımlamadığınız takdirde <mod> = R olarak okumak için açılır. Dosya <tip>'i var olan bir relative (göreli) dosyayı açmak (OPEN) için kullanılır. Relative dosyaları kullanmak için <tip>'i REL olarak tanımlayın. Relative ve sequential dosyalar yalnızca disk için kullanılır.

Önceden OPEN deyimi ile açılmamış olan bir dosyaya erişmek istediğinizde **FILE NOT OPEN** (dosya açık değil) hatası oluşur. Var olmayan bir dosyayı okumak için açmaya çalıştığınızda ise bu, **?FILE NOT FOUND** (dosya bulunamadı) hatasına neden olacaktır. Diskete yazmak için, var olan bir dosya ismi ile dosya açmaya çalıştığınızda **FILE EXIST** (dosya zaten var) DOS hatası oluşur. Teypteki dosyalar için bu tür hataların oluşmasını önlemek olanağı yoktur. Bu yüzden de kaydedilecek verilerin daha önce kaydedilmiş olan verilerin üzerine gelmemesi için kasetin uygun konuma getirildiğine emin olmanız gerekir. O anda açık bulunan bir dosyayı tekrar açmaya çalıştığınızda oluşacak hata ise. **FILE OPEN** (dosya açık) hatasıdır. (Daha fazla ayrıntı için yazıcı el kitabına bakınız.)

```
10 OPEN 2, 8, 4 "DISK-CIKISI,SEQ,W"
```

(Disket üzerinde sequential dosya açar.)

```
10 OPEN 1, 1, 2, "TEYP-YAZMA"
```

(Teyp üzerine yazmayı kapatırken dosya-sonu yazar.)

```
10 OPEN 50, 0
```

(Klavyeden giriş.)

```
10 OPEN 12, 3
```

(Ekrana çıkış.)

```
10 OPEN 130, 4
```

(Yazıcıya çıkış.)

```
10 OPEN 1, 1, 0, "ISIM"
```

(Kasetten okuma.)

```
10 OPEN 1, 1, 1, "ISIM"
```

(Kasete yazma.)

```
10 OPEN 1, 2, 0, CHR$(10)
```

(RS-232 cihazı için kanal açmak.)

```
10 OPEN 1, 4, 0, "YAZINSAL DIZI"
```

(Yazıcıya büyük harf/grafik göndermek.)

```
10 OPEN 1, 4, 7, "YAZINSAL DIZI"
```

(Yazıcıya büyük/küçük harf göndermek.)

```
10 OPEN 1, 5, 7, "YAZINSAL DIZI"
```

(5 nolu yazıcıya büyük/küçük harf göndermek.)

```
10 OPEN 1, 8, 15, "KOMUT"
```

(Diske komut göndermek.)

OR (OR: Veya) (Mantıksal operatör)

<ifade> OR <ifade>

Bağıntı operatörlerinin, programın akışına göre karar alınması için kullanıldığı gibi, mantıksal operatörler de bir ya da daha fazla bağıntıyı birleştirmek ve daha sonra kararlarda kullanılacak olan doğru ya da yanlış şekildeki değerleri elde etmek için kullanılırlar.

Hesaplamalarda kullanıldıkları zaman kıyaslanan bitlerin herhangi birinin ya da ikisinin birden 1 olduğu durumlarda OR işleminin sonucu 1'dir. Bu işlem sonucunda, ifadelerin değerine bağlı olarak tamsayı bir değer üretecektir. Kıyaslamalarda kullanıldığında ise iki ifadeyi tek bir ifadeye dönüştürmek için (bu iki ifadeyi birbirine bağlamak için) kullanır. Eğer her iki ifade de doğru ise, birleştirilmiş ifadenin değeri -1'dir. Aşağıdaki ilk örnekte AA, BB'ye eşitse ya da <OR> eğer XX = 20 ise, ifade doğrudur.

Mantıksal operatörlerde, ifadeler -32768 ile +32767 arasında olması gerekir. İfadeler -32768 ile +32767 arasında değilse bir hata mesajı alınır.

```
100 IF (AA=BB) OR (XX= 20) THEN END
```

```
230 KK%=64 OR 32: PRINT KK%
RUN
 96

READY.
```

PEEK (PEEK: Bellek yerleşimini oku) (Sayısal fonksiyon)

PEEK (<sayısal>)

İşleme sokulduğunda, bellek yerleşiminden okuduğu 0 ile 255 arasında yer alan bir sayıyı verir. <sayısal> deyimi, 0 ile 65535 arasındaki bir bellek adresidir. Eğer sayı 0'dan küçük veya 65535'den büyük verilirse **? ILLEGAL QUANTITY** (geçersiz miktar) hatası oluşur.

```
10 PRINT PEEK(53280) AND 15
```

(Ekranın sınır renginin değerini verir.)

```
5 A%=PEEK(45)+PEEK(46)*256
```

POKE **(POKE: Bellek yerleşimine yaz) (Yönerge)**

POKE <yerleşim>, <değer>

POKE yönergesi, verilmiş olan bellek <yerleşim>'ine veya giriş-çıkış register'ine, bir baytlık (8-bit) ikili <değer> yazmak için kullanılır. <yerleşim> aritmetiksel bir ifadedir ve 0 ile 65535 arasında olmak zorundadır. <değer> ise 0 ile 255 arasında yer alan tamsayı bir ifadedir. Her iki ifade de, verilen sınırlar içinde olmak zorundadır yoksa **?ILLEGAL QUANTITY** hatası oluşur.

POKE ve PEEK yönergeleri, bellek yerleşimleri ile ilgili tanımlı fonksiyonlardır ve veri saklamada, grafiklerin ses ve görüntü yaratımlarını kontrol etmede, assembly dilindeki alt-programları yüklemede, argümanların ve sonuçların assembly dilinde alt-programlara geçirilmesinde ve alınmasında oldukça faydalıdır. Ek olarak, işletim sisteminin değişkenleri, PEEK yönergesi ile incelenebilir. POKE yönergesi ile de değiştirilebilir ya da işlenebilirler. Kullanıma elverişli yerleşimleri de içeren komple bir bellek haritası. EK-G'de verilmiştir.

```
POKE 1024, 1
```

(Ekrandaki 1'inci pozisyona "A" koyar.)

```
POKE 2040, PTR
```

(Yaratık no. 0 veri göstergecini yeniler.)

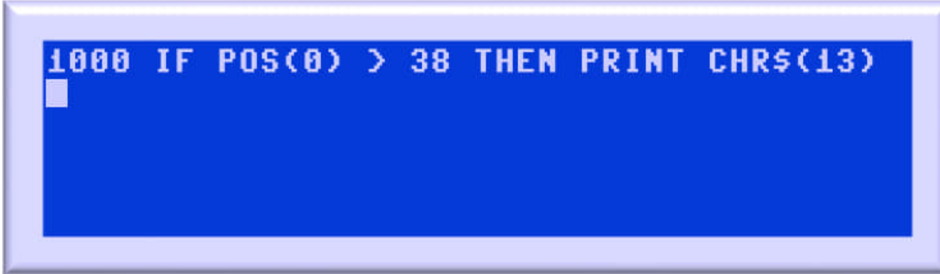
```
10 POKE RED, 32
20 POKE 36879, 8
```

```
2050 POKE A, B
```

POS (POSITION: Konum) (Sayısal fonksiyon)

POS (<boş değer>)

80 karakterlik mantıksal ekran satırında, takipçinin o anda bulunduğu konumu verir. Bu konum, 0 (en soldaki karakter) ile 79 arasında bir yerdir. Commodore 64'ün ekranı 40 sütunluk olduğundan 40 ile 79 arasındaki bir konum, 2'nci satırda yer alacaktır. Parantez içinde verilen sayı, sonucu etkilemez.



PRINT (PRINT: Yaz) (Yönerge)

PRINT [<değişken>] [<,/;><değişken>] …

PRINT yönergesi genelde, verileri ekranda görüntülemek için kullanılır. CMD yönergesi ise çıkışın, sistemdeki başka bir cihaza yöneltmesini sağlar. Çıktı listesindeki <değişken> ya da değişkenler, herhangi bir tipteki ifadelerdir. Çıktı listesi yok ise, boş bir satır basılacaktır. Basılan değişkenlerin konumu, çıktı listesinde değişkenleri birbirinden ayırmak için kullanılan noktalama işaretleri tarafından belirlenir.

Kullanabileceğiniz noktalama işaretleri ise virgül, noktalı virgül veya boşluklardır. 80 karakterlik mantıksal ekran satırı her biri 10 karakter uzunluğundaki 8 PRINT bölgesinden oluşur. İfadeler listesinde kullanacağınız bir virgül bir sonra gelen değerin, bir sonra gelen bölgenin başlangıcında basılmasına neden olur. Noktalı virgül ise, bir sonra gelen değerin, basılan değerin hemen yanına basılmasını sağlar. Fakat bu kurala uymayan iki tane istisna vardır:

1. Sayısal değerlerin arkasına bir boşluk eklenir.
2. Pozitif sayılardan önce de bir boşluk basılır.

Yazınsal dizi sabitleri ve değişken isimleri arasında noktalama işareti kullanmadığınızda veya boşluk koyduğunuzda, noktalı virgül kullanılmış gibi olur. Fakat bir yazınsal dizi bir sayısal değer veya iki sayısal dizi arasında boşluk kullandığınızda, ikinci değer basılmadan çıkış durdurulacaktır.

Eğer bir virgül veya bir noktalı-virgül çıktı listesinin sonunda yer alıyorsa, bir sonraki PRINT yönergesi, yazımın kaldığı yerden başlar ve yerine göre boşluk bırakılır. Listenin sonunda noktalama işareti yoksa verinin sonunda bir RETURN ve satır başı işareti basılır. Bir sonraki PRINT yönergesi işlemini bir sonraki satırda sürdürecektir. Eğer çıktınız ekrana yönelikse ve veri 40 sütundan uzun ise, çıktı bir sonraki ekran satırında devam edecektir.

BASIC'te PRINT yönergesi kadar fazla çeşitleme yapma olanağı olan başka bir yönerge yoktur. Bu yönergeyi, BASIC içinde, ekran olaylarını düzenleyen apayrı bir mini bilgisayar-dili olarak niteleyebiliriz.

```
5 X = 5
10 PRINT -5*X, X-5, X+5, X↑5
RUN
-25         0          10         3125

READY.
```

```
5 X = 9
10 PRINT X; "UN KARESI";X*X;"VE";
20 PRINT X "UN KUBU" X↑3
RUN
 9 UN KARESI 81 VE  9 UN KUBU  729

READY.
```

```
90 AA$="ACABA":BB$="BUGUN":CC$="NASILSIN
IZ":DD$="BEN": EE$="MERAKTAYIM"
100 PRINT AA$BB$;CC$ DD$,EE$
RUN
ACABABUGUNNASILSINIZBEN       MERAKTAYIM

READY.
```

Tırnak işareti modu:

Tırnak işareti **SHIFT** **2** girildiğinde, takipçi tuşları işlevini yapmaz ve bu tuşlara bastığınızda ekranda tersine grafik işaretleri görünür. Bu size, bu tuşları BASIC programınızın kontrolünde kullanma imkânı tanır. **INST/DEL** tuşu tırnak işareti modundan etkilenmeyen tek takipçi tuşudur.

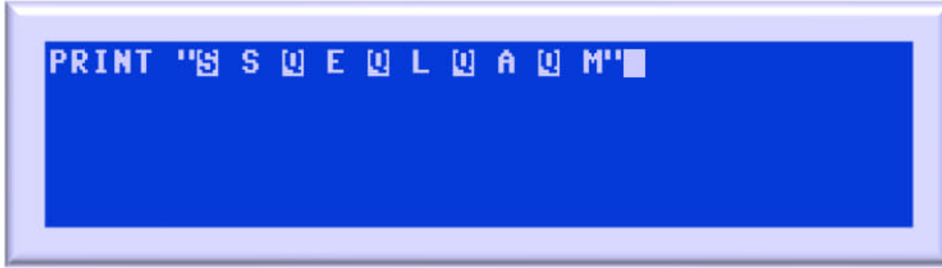
1. Takipçi hareketi

Tırnak işareti modunda "programlanmış" takipçi kontrolleri şunlardır.

KONTROL TUŞU	EKRAN GÖRÜNTÜSÜ
CLR/HOME	
SHIFT CLR/HOME	
↑CRSR↓	
SHIFT ↑CRSR↓	
←CRSR→	
SHIFT ←CRSR→	

Eğer SELAM sözcüğünü ekranın sol üst köşesinden başlayıp aşağı doğru yazdırmak istiyorsanız:

PRINT "CLR/HOME S ↑CRSR↓ E ↑CRSR↓ L ↑CRSR↓ A ↑CRSR↓ M" yazdığınızda, ekrandaki görüntüsü:



2. Ters Karakterler

CTRL tuşunda parmağınızı basılı tutarak 9 'a bastığınızda tırnak işareti içinde 'nin yer aldığını görürsünüz. Bu, karakterlerin reverse video dediğimiz bir resmin negatifi şeklinde görüntülenmesini sağlar. Zıt renkteki görüntülenmeyi kesmek için, görüntüsü olan CTRL 0 tuşlarına veya RETURN (CHR$(13))'e basmanız gerekir (tırnağı kapatmadan önce).

3. Renk Kontrolü

Parmağınız CTRL ya da C= tuşunun üzerindeyken, 8 renk tuşundan herhangi birine bastığınızda tırnak içinde özel bir tersine karakter görüntülenir. Karakter PRINT edildiğinde, renk değişikliği oluşacaktır.

KONTROL TUŞU	RENK	EKRAN GÖRÜNTÜSÜ
CTRL 1	Siyah	
CTRL 2	Beyaz	
CTRL 3	Kırmızı	
CTRL 4	Turkuvaz	
CTRL 5	Eflatun	
CTRL 6	Yeşil	
CTRL 7	Mavi	
CTRL 8	Sarı	
C= 1	Turuncu	
C= 2	Kahverengi	
C= 3	Açık kırmızı	
C= 4	Gri 1	
C= 5	Gri 2	
C= 6	Açık yeşil	
C= 7	Açık mavi	
C= 8	Gri 3	

Eğer SELAM sözcüğünü sarı, ORADAKILER sözcüğünü de beyaz olarak bastırmak istiyorsanız:

PRINT " CTRL 8 SELAM CTRL 2 ORADAKILER" yazın RETURN basın görüntü aşağıdaki gibi olacaktır:

```
PRINT "█SELAM █ORADAKILER"
SELAM ORADAKILER

READY.
█
```

4. Insert Modu

INST/DEL tuşuna basarak açacağınız boş yerler, tırnak işaretinin içindeymiş gibi davranacaklardır. Takipçi ve renk kontrol karakteri tersine yazımda gösterilecektir. Tek farklı işlev gören tuş, INST/DEL tuşu olacaktır. Tırnak içinde normal işlevini gören DEL, artık tersine T harfini oluşturacaktır. Tırnak içinde tersine karakter oluşturan INST ise, karakterlerin arasını açmaya başlayacaktır.

İşte bu özellik, program çalıştığında ekrana yazılan yazıları, gene program aracılığıyla silmeyi olanaklı kılar, tıpkı DEL tuşuna basılmış gibi. İşte örnek:

10 PRINT "SULAK" INST/DEL SHIFT INST/DEL SHIFT INST/DEL INST/DEL INST/DEL H" ve ekrandaki görüntüsü de aşağıdaki gibi:

```
10 PRINT "SULAK██H"
RUN
SULH

READY.
█
```

Yukarıdaki örnekte göründüğü üzere sonuç "SULH" olacaktır. Burada, RUN yaptığınızda, ekrana önce SULAK yazılacak, son iki harf otomatik olarak silinecek ve yerine H yazılarak SULH (barış) kelimesi elde edilecektir.

DİKKAT: 10 numaralı satırı LIST edip değiştiremezsiniz. Satırı yeniden yazmanız gerekir.

Insert modu RETURN (veya SHIFT RETURN) tuşuna basıldığında veya açılan boşluk kadar yazıldığında sona erer.

5. Diğer özel karakterler

Klavyeden girilmesi kolay olmayan, fakat özel amaçlar için basılması (PRINT) gereken bazı karakterler vardır. Bunları tırnak işaretleri içinde yazmak yerine, satırda bunlar için boş yerler bırakmalı, **RETURN** ya da **SHIFT** **RETURN** tuşuna basmalı ve takipçi kontrol tuşlarını kullanarak boş bırakılan yerlere geri dönmelisiniz. Şimdi de **CTRL** **RVS/ON** tuşlarına aynı anda basın ve tersine karakterleri yazmaya başlayın. Aşağıdaki tuşları girebilirsiniz:

FONKSİYON	KONTROL TUŞU	EKRAN GÖRÜNTÜSÜ
SHIFT RETURN	SHIFT M	
Küçük harfe geçiş	N	
Büyük harfe geçiş	SHIFT N	
Büyük/küçük harf değiştirme tuşlarını devre dışı bırakma	H	
Büyük/küçük harf değiştirme tuşlarını	I	

SHIFT **RETURN** tuşu, LIST ve PRINT için aynı çalışır. Eğer bu karakter kullanılmış ise sonradan üzerinde değişiklik yapmak imkansızdır. LIST edildiğinde garip şeyler görürsünüz.

PRINT# (PRINT#: Dosyaya yaz) (G/Ç yönergesi)

PRINT# <dosya-no> [<değişken>] [<,/;><değişken>] …

PRINT# yönergesi, verileri, bir mantıksal dosyaya kaydetmek amacıyla kullanır. <dosya-no>, dosyayı açmak için OPEN yönergesinde kullanılan <dosya-no> ile aynı olmak zorundadır. Çıktı, OPEN yönergesinde kullanılmış olan <cihaz-numarası> numaralı cihaza yöneltilir. Çıktı listesindeki <değişken> ifadeler, herhangi bir türde olabilirler. Birimler arasındaki noktalama işaretleri, PRINT yönergesindekilerle aynıdır ve aynı şekilde kullanılabilirler. Noktalamanın etkileri iki önemli açıdan farklıdır.

PRINT#, teypteki dosyalar için kullanıldığında, virgül, basılacak birimler arasında boşluk bırakmak yerine, noktalı virgül gibi işlem görür. Bu yüzden birimler arasına yerleştirilecek: boşluklar, virgüller, noktalı virgüller ya da diğer noktalama işaretleri aynı boşluk bırakılmış gibi işlem görürler. Veri birimleri, sürekli bir karakter akışı şeklinde yazılırlar. Sayısal birimlerin ardından bir boşluk gelir. Eğer pozitif ise, bir de önüne boşluk konulur.

Eğer listenin sonunda herhangi bir noktalama işareti yoksa, verilerin sonuna, bir RETURN ya da satırbaşı yazılır. Çıktı-listesinin sonunda bir virgül ya da bir noktalı virgül yer alıyorsa CARRIAGE-RETURN ve LINE-FEED basılır. Noktalama işaretlerini hesaba katmaksızın, bir sonraki PRINT# yönergesi bir sonra gelen elverişli karakter pozisyonundan çıktı vermeye başlayacaktır. INPUT# yönergesi içinde kullanıldığında LINE-FEED. INPUT# işlendiği zaman bir boş değişken bırakarak bir stop olarak hareket edecektir.

Teypteki ya da diskteki bir dosyaya birden fazla değişken yazabilmenin en kolay yolu, bir yazınsal dizi değişkenini CHR$(13)'e eşitlemek ve dosyayı yazarken bu yazınsal diziyi diğer değişkenlerin arasında kullanmaktır.

```
10 OPEN 1, 1, 1, "TEYP DOSYASI"
20 R$ = CHR$(13)
30 PRINT# 1,1;R$;2;R$;3;R$;4;R$;5
40 PRINT# 1,6
50 PRINT# 1,7
```

CHR$(13)'ü CHR$(44) olarak değiştirirseniz her değişkenin arasına "," koymuş olursunuz. CHR$(59) ile ise değişkenlerin arasına ";" konacaktır.

```
10 CO$=CHR$(44):CR$=CHR$(13)
20 PRINT#1, "AAA" CO$ "BBB", "CCC";"DDD"
;"EEE"CR$"FFF",CR$;
30 INPUT#1,A$,BCDE$,F$
```

Satır 20'de AAA,BBB CCCDDDEEE (carriage return) FFF (carriage return)

```
5 CR$=CHR$(13)
10 PRINT#2, "AAA";CR$;"BBB"
20 PRINT#2, "CCC";
30 INPUT#2, A$,B$,DUMMY$,C$
```

Satır 10'da (10 boşluk) AAA BBB
Satır 20'de (10 boşluk) CCC

READ (READ: Oku) (Yönerge)

READ <değişken> [, <değişken>] …

READ yönergesi ile değişken isimler, DATA yönergelerinde verilmiş olan sabit değerlerle doldurulur. Okunacak olan veri, tanımlanmış olan değişkenle aynı tipte olmak zorundadır yoksa **?SYNTAX** hatası oluşur. (**SYNTAX** yani yazım hatası DATA yönergesinin bulunduğu satır için verilecektir, READ yönergesinin bulunduğu satır için değil). DATA ile verilen değişkenler birbirlerinden virgülle ayrılmak zorundadır.

Tek bir READ yönergesi, bir ya da sıraya uyacak şekilde birden fazla DATA yönergesine erişebilir (DATA yönergesine bakınız). Ya da bir sürü READ yönergesi aynı DATA yönergesini kullanabilirler. Eğer programda DATA yönergesinde verilen değişkenlerin sayısından fazla olacak şekilde okuma yapmaya çalışılırsa, **?OUT OF DATA** hatası oluşur. Eğer verilen değişkenlerin sayısı DATA yönergesindeki elemanların sayısından az ise, bir sonra gelen READ yönergesi okumaya, son okunan elemanın yanındaki elemandan başlayarak devam edecektir (RESTORE yönergesine bakınız).

NOT: **?SYNTAX ERROR**, READ deyiminde değil, DATA deyimindeki satır numarasıyla birlikte görünecektir.

```
110 READ A, B, C$
120 DATA 1, 2, SELAM
```

```
100 FOR X=1 TO 10: READA(X): NEXT
200 DATA 3.08, 5.19, 3.12, 3.98, 4.24
210 DATA 5.08, 5.55, 4.00, 3.16, 3.37
```

```
1 READ IL$, KAZA$, PKOD
5 DATA ORDU, FATSA, 52400
```

(Dizi öğelerini (1. satır) gösterilen sabitlerin sırasına göre doldurur (satır 5))

REM (REMARK: Hatırlatma) (Yönerge)

REM [<hatırlatma>]

REM yönergeleri, listesi alınan programların daha kolay anlaşılmasını sağlarlar. Programın bölümlerini yazarken kafanızdan neler geçtiğini hatırlatmaya yarar. Örneğin; değişkenleri ne için kullandığınızı ya da diğer faydalı bilgileri sonradan hatırlamak isteyebilirsiniz. REMARK yani uyarı, herhangi bir yazı, bir sözcük ya da (:) içeren bir karakter ya da bir BASIC anahtar sözcüğü olabilir.

REM yönergeleri ve aynı satırda ondan sonra gelen her şey BASIC tarafından göz ardı edilir. Fakat REMark'lar, programın listesini aldığınızda, girdiğiniz şekilde görüntülenirler. REM yönergesine GOTO ya da GOSUB yönergesi ile ulaşabilirsiniz. Programın işletilmesi ise, işlenebilecek yönergenin bulunduğu bir sonraki daha büyük satır numarasından başlayarak devam edecektir.

```
10 REM ORTALAMA HIZI HESAPLAMA
20 FOR X=1 TO 20: REM 20 DEGER ICIN BIR
DONGU
30 SUM=SUM + VEL(X): NEXT
40 AVG=SUM/20
```

RESTORE (RESTORE: Yenile) (Yönerge)

RESTORE

BASIC'te, DATA yönergesinde verilen bir sonraki veriyi okumak için (READ yönergesi ile), bir iç göstergeci bulunur. Programın içinde kullanılacak bir RESTORE yönergesi ile bu göstergecin tekrar DATA yönergesindeki ilk veriyi göstermesi sağlanabilir. RESTORE yönergesi, DATA yönergesinde verilen verilerin tekrar okunması için programın herhangi bir yerinde kullanılabilir.

```
100 FOR X=1 TO 10: READ A(X): NEXT
200 RESTORE
300 FOR Y=1 TO 10: READ B(Y): NEXT
4000 DATA 3.08, 5.19, 3.12, 3.98, 4.24
4100 DATA 5.08, 5.55, 4.00, 3.16, 3.37
```

(İki dizeyi de aynı verilerle doldurur.)

```
10 DATA 1,2,3,4
20 DATA 5,6,7,8
30 FOR L=1 TO 8
40 READ A: PRINT A
50 NEXT
60 RESTORE
70 FOR L=1 TO 8
80 READ A: PRINT A
90 NEXT
```

RETURN (RETURN: Geri dön) (Yönerge)

RETURN

RETURN yönergesi, GOSUB yönergesi ile çağırılan bir alt-programdan çıkıp, geriye dönmek için kullanılır. RETURN, GOSUB'dan sonra gelen bir sonraki işlenebilir yönergeden itibaren programın geri kalan kısmını işletir. İç içe geçmiş alt programlarda, her GOSUB yönergesine karşılık gelen, en az bir RETURN yönergesi olmak zorundadır. Bir alt program birçok sayıda RETURN yönergesi içerebilir, fakat ilk erişilen, işleme sokulur ve alt programdan çıkılmasını sağlar.

```
10 PRINT "BU PROGRAM"
20 GOSUB 1000
30 PRINT "PROGRAM DEVAM EDIYOR"
40 GOSUB 1000
50 PRINT "BASKA PROGRAM"
60 END
1000 PRINT "BU ALT-PROGRAM": RETURN
```

RIGHT$ (RIGHT: Sağ) (Yazınsal dizi fonksiyonu)

RIGHT$ (<yazınsal-dizi>, <sayısal>)

RIGHT$ fonksiyonu, <yazınsal-dizi>'de verilen argümanından en sağdan başlayarak, başa doğru giden bir alt-dizi elde etmenizi sağlar. Alt- dizinin uzunluğu <sayısal>'da verilen değer ile belirlenir ve bu değer 0 ile 255 arasında bir tam sayı olmalıdır. Eğer sayısal ifadenin değeri 0 ise, sonuç boş dizi yani ("")'dır, ve değer <yazınsal-dizi>'nin uzunluğundan büyük ise dizinin tamamı elde edilir.

```
10 MSG$ ="COMMODORE BILGISAYARLARI"
20 PRINT RIGHT$(MSG$,14)
RUN
BILGISAYARLARI

READY.
```

RND (RANDOM: Rastlantısal) (Sayısal fonksiyon)

RND (<sayısal>)

RND, 0.0 ile 1.0 arasında tesadüfi bir sayı üretir. Bilgisayar, bu sayıyı üretmek için daha önceden belirlenmiş bir sayı üzerinde bazı işlemler yapar. Bu başlangıç değeri, bilgisayarı açtığınız anda yerine yerleştirilir. <sayısal> için vereceğiniz değerin sadece sıfır, pozitif veya negatif olması önemlidir; sayı değerinin bir anlamı yoktur.

Eğer <sayısal> pozitif bir sayı ise, arka arkaya üretilecek olan tesadüfi sayılar bir noktadan sonra kendi kendilerini tekrar etmeye başlayacaklardır. Aynı başlangıç değerinin, aynı sayı dizisini oluşturacağını bilmek, programları kontrol etmekte yararlı olabilir.

Eğer <sayısal> sıfır ise, bu durumda RND gereken sayıyı, bilgisayarın içinde bulunan saat yardımıyla üretecektir.

Son olarak, negatif bir <sayısal> RND fonksiyonunun başlangıç değerini yeniden ilk değer haline getirecektir.

```
220 PRINT INT(RND(0)*50)
```

(0-49 arası rastgele üretilen tam sayılar.)

```
100 X=INT(RND(1)*6)+INT(RND(1)*6)+2
```

(İki zarın toplamını verir.)

```
100 X=INT(RND(1)*1000)+1
```

(1-1000 arası rastgele üretilen tam sayılar.)

```
100 X=INT(RND(1)*150)+100
```

(100-249 arası rastgele üretilen tam sayılar.)

```
100 X=INT(RND(1)*150)+100
```

(U-L ile gösterilen sınırlar arasında, rastgele üretilen tam sayılar.)

RUN **(RUN: Çalıştır) (Komut)**

RUN [<satır-no>]

Sistem komutu olan RUN, o anda bellekte yer alan programı başlatmak için kullanılır. RUN komutu, programın başlamasından önce CLR işleminin yerine getirilmesine neden olur. Silinme (Clearing) işleminden kaçınmak için RUN yerine CONT veya GOTO kullanmak gereklidir. Eğer <satır-no> verilmişse, program bu satırdan başlayarak işletilecektir, diğer durumlarda RUN komutu programın ilk satırından başlar.

RUN komutunu programın içinde de kullanmak mümkündür. Eğer verilen <satır-no> programın içinde yer almıyorsa, yani bu numarada herhangi bir satır yoksa, UNDEFINED STATEMENT hatası oluşur.

İşlemekte olan bir program STOP veya END yönergesine eriştiğinde, programın son satırı da işlediğinde veya işletim sırasında BASIC hatası oluştuğunda durur ve BASIC doğrudan moda (direct mode) geri döner.



(Program ilk satırdan başlar.)



(Program 500. satırdan başlar.)



(Program 1000. satırdan başlar. Eğer 1000. satır yoksa **UNDEF'D STATEMENT ERROR** hatası olur.)

SAVE (SAVE: Kaydet) (Komut)

SAVE ["<dosya-ismi>"] [,<cihaz-no>] [,<adres>]

SAVE komutu, o anda bellekte bulunan programı kasete veya diskete kaydetmek için kullanılır. SAVE komutu ile kaydedilmekte olan program, komut tarafından yalnızca saklama işlemi sırasında etkilenir. SAVE işlemi bittikten sonra eğer siz başka komutlar kullanarak yeni şeyler oluşturmadıysanız, program bilgisayarın belleğinde olduğu gibi kalır. Dosya tipi "prg" (program) olacaktır. Eğer <cihaz-no> belirtmediyseniz, C-64 otomatik olarak sizin programınızı cihaz numarası 1 olan teybe kaydetmek istediğinizi düşünür. <cihaz-no> olarak <8> verdiyseniz programınız diskete saklanacaktır.

SAVE komutunu programlarınızın içinde de kullanabilirsiniz ve SAVE işlemi tamamlandıktan sonra programınız bir sonraki komuta geçerek işlemeye devam edecektir.

Kaset üzerindeki programlar otomatik olarak iki kez kaydedilir, böylece Commodore 64'ünüz programı tekrar yüklediğinizde (LOAD ile) hataları kontrol edebilir. Programların kasete kaydedilmesi sırasında da, <dosya-ismi> ve ikincil <adres>'in kullanımı isteğe bağlıdır. Fakat SAVE komutunun ardından tırnak içinde ("") veya yazınsal dizi (...$) halinde vereceğiniz bir program ismi Commodore 64'ün programları kolaylıkla bulmasını sağlayacaktır. Eğer başta <dosya-ismi> vermediyseniz, sonradan isim vererek yüklemeniz (LOAD) mümkün değildir.

İkincil adres 1, KERNAL'a normalde 2048 olan bellek adresi (programın başladığı) yerine o anda bellekte yer alan programın gerçek başlangıç adresinin kasete kaydedilmesini bildirir (makina dili programları için). İkincil adres 2 ise, programın sonuna kaset sonu işareti koyulmasını sağlar. İkincil adres 3 bu iki fonksiyonu birleştirir.

Programları diskette kaydetmek isterseniz. <dosya-ismi> vermeniz zorunludur.

```
SAVE
```

(Teyp kasete isimsiz olarak kaydeder.)

```
SAVE "HESAP", 1
```

(Teyp kasete <dosya-ismi> HESAP olarak kaydeder.)

```
SAVE "HESAP", 1, 2
```

(Teyp kasete <dosya-ismi> HESAP olarak kaydeder ve dosya sonu işareti koyar.)

```
SAVE "HESAP", 8
```

(8 nolu cihazda diskete <dosya-ismi> HESAP olarak kaydeder.)

```
SAVE A$
```

(Teyp kasete A$'daki <dosya-ismi> ile kaydeder.)

```
10 SAVE "HESAP"
```

(Teyp kasete <dosya-ismi> HESAP olarak kaydeder ve varsa bir sonraki satırdan programı işletmeye devam eder.)

SGN (SIGNE: İşaret) (Sayısal fonksiyon)

SGN (<sayısal>)

SGN, <sayısal> değerin işaretine bağlı olarak tamsayı bir değer verir. Eğer <sayısal> değişken pozitif ise sonuç 1, sıfır ise sonuç 0, negatif ise sonuç -1'dir.

```
90 ON SGN(DV)+2 GOTO 100, 200, 300
```

(DV= negatif ise 100'e, DV=0 ise 200'e, DV= pozitif ise 300'e dallanır.)

SIN (SINUS: Sinüs) (Sayısal fonksiyon)

SIN (<sayısal>)

SIN <sayısal> değişkenin sinüsünü radyan cinsinden verir. COS(X)'in değeri SIN(X +3.14159265/2)'e eşittir.

```
235 AA=SIN(1.5): PRINT AA
RUN
 .997494987

READY.
```

SPC (SPACE: Boşluk) (Özel fonksiyon)

SPC (<sayısal>)

SPC fonksiyonu, ekrana ya da mantıksal bir dosyaya yapılacak çıkışlarda, verilerin formatlanmasını kontrol etmek için kullanılır. <sayısal> değişkende belirtilen sayıda elverişli olan ilk konumdan başlayarak boşluk (space) bırakılır. Ekran ya da kasetteki dosyalar için <sayısal> değişken 0'la 255 arasında, disketteki dosyalar için 254'e kadar olan bir değerdir. Yazıcı dosyalarında eğer satırın en sonundaki boşluk bastırılıyorsa, yazıcı otomatik olarak satır-başı yapar ve alt satıra geçer. Aşağıdaki satırda boşluk basılmaz.

```
10 PRINT "KALEM "; "BURADA";
20 PRINT SPC(5) "SILGI" SPC(11) "ORADA"
RUN
KALEM BURADA     SILGI           ORADA

READY.
```

SQR (SQUARE: Karakök) (Sayısal fonksiyon)

SQR (<sayısal>)

<sayısal> değişkende verilen değerin kare kökünü verir. Değişken negatif bir değer olmamalıdır. Bu **?ILLEGAL OUANTITY** hatasına neden olur.

```
FOR J=2 TO 5: PRINT J*5, SQR(J*5): NEXT
 10        3.16227766
 15        3.87298335
 20        4.47213595
 25        5

READY
```

STATUS (STATUS: Durum) (Sayısal fonksiyon)

STATUS

Açık bir dosyada yapılan, son Giriş/Çıkış işlemi için tamamlama DURUM'unu (STATUS) verir. STATUS çevre cihazlarının herhangi bir tanesinden okunabilir. STATUS <ya da daha basit gösterimle ST) anahtar sözcüğü, KERNAL'ın Giriş/Çıkış işlemlerinin DURUMU'nu (STATUS) yerleştirdiği sistem tarafından tanımlanabilir değişken ismidir. Teyp, yazıcı, disk ve RS-232 dosya işlemleri için STATUS kod değerleri aşağıdaki tabloda verilmiştir.

ST Bit Konumu	ST Sayısal Değeri	Kaset Okuma	Seri Yol Okuma/Yazma	Teyp Kontrol + Yükleme
0	1		Yazma Zaman Aşımı	
1	2		Okuma Zaman Aşımı	
2	4	Kısa Blok		Kısa Blok
3	8	Uzun Blok		Uzun Blok
4	16	Anlaşılmayan Okuma Hatası		Herhangi Bir Uyuşmazlık
5	32	Sağlama Toplamı Hatası		Sağlama Toplamı Hatası
6	64	Dosya Sonu	Kesinti Sonu	
7	128	Kaset Sonu	Cihaz Hazır Değil	Kaset Sonu

```
10 OPEN 1, 4: OPEN 2, 8, 4, "ANA DOSYA
SEQ,W"
20 GOSUB 100: REM STATUS'U KONTROL ET
30 INTPUT# 2, A$, B, C
40 IF STATUS AND 64 THEN 80: REM DOSYA S
ONUNU ARASTIR
50 GOSUB 100: REM STATUS'U KONTROL ET
60 PRINT# 1, A$, B; C
70 GOTO 20
80 CLOSE 1: CLOSE 2
90 GOSUB 100: END
100 IF ST > 0 THEN 9000: REM DOSYA GIRIS
/CIKIS HATASI VAR MI
110 RETURN
```

STEP (STEP: Adım) (Yönerge)

[STEP <ifade>]

STEP, FOR yönergesi içinde <son-değer>'den sonra yer alan keyfi bir anahtar sözcüktür. Döngü içinde, artış değerini belirtir. STEP ile herhangi bir değerin kullanılması mümkündür, yalnız sıfır değerinin sonsuz döngüye neden olacağını unutmayın. STEP sözcüğü kullanılmamışsa artış değeri +1 olacaktır. FOR döngüsü içinde NEXT yönergesine gelindiğinde <son-değer>'e eşit olana kadar STEP ile verilen sayıda artmaya devam eder. <son-değer>'e eşit olduğunda döngü sona erer. (Daha fazla bilgi için FOR yönergesine bakın.)

NOT: Döngü içinde STEP değerini değiştirmek mümkün değildir.

```
10 FOR XX = 2 TO 20 STEP 2
```

(Döngü 10 kez tekrarlanır.)

```
10 FOR ZZ = 0 TO -20 STEP -2
```

(Döngü 11 kez tekrarlanır.)

STOP (STOP: Dur) (Yönerge)

STOP

STOP yönergesi o anda işlemekte olan programı durdurup doğrudan moda, (direct mode) dönmek için kullanılır. STOP yönergesi, **RUN/STOP** tuşuna basmakla aynı işi görür. BASIC hata mesajı, **BREAK IN SN** (Satır "SN"'de bırakıldı) ekranda görüntülenir. "SN", STOP'un sebep olduğu kesintinin satır numarasıdır. Açık dosyalar bırakılma sırasında açık kalır ve dosyadaki değişkenlerin değeri korunur ve bu anda incelenebilir. Program, CONT veya GOTO yönergeleri ile tekrar başlatılabilir.

```
10 INPUT# 1, AA, BB, CC
20 IF AA = BB AND BB = CC THEN STOP
30 STOP
```

(AA'nın değeri -1 ise ve BB, CC'ye eşit ise STOP'un neden olduğu hata **BREAK IN 20**, bunun dışındaki değerlerde **BREAK IN 30** olur.)

STR$ (STRING: Yazınsal dizi) (Yazınsal dizi fonksiyon)

STR$ (<sayısal>)

<sayısal>'da verilen değerin yazınsal dizi şeklindeki halini verir. STR$ değeri, <sayısal> argümanda verilmiş olan değişkenlere dönüştürüldüğünde, sayılar arkalarına bir boşluk ekleyerek ve eğer sayı pozitif ise önüne de bir boşluk ekleyerek gösterilir.

```
100 FLT =1.5E4: ALFA$ = STR$(FLT)
110 PRINT FLT, ALFA$
RUN
 15000     15000

READY.
```

SYS (SYSTEM: Sistem) (Yönerge)

SYS <bellek-yerleşim>

Bu, BASIC dilinde yazılmış olan bir programı, makina dilinde yazılmış bir programla karıştırmak için çok sık kullanılan bir yoldur. Makina dilindeki program SYS yönergesinde verilen <bellek-yerleşim>'inden başlar. SYS komutu doğrudan (direct) ya da program modunda kullanılarak, mikroişlemcinin kontrolünün, bellekte yer alan makina dili programına geçirilmesi sağlanır. Sayısal ifade olarak verilen <bellek-yerleşim>'i RAM veya ROM'un herhangi bir yerinde olabilir.

SYS yönergesini kullanırken, makina dili kodu bölümünü bir RTS (Alt-programdan geriye dön) yönergesiyle bitirmeniz gerekir. Böylece makina dili programı sona erdiğinde BASIC, SYS komutundan sonra gelen yönergeye geçerek işlemeye devam edecektir.

```
SYS 64738
```

(ROM'saki sistem başlangıcına atlar.)

```
10 POKE 4400,96: SYS 4400
```

(Önce 4400'e RTS'nin kodunu yerleştirir. SYS'den sonra da, RTS nedeniyle hemen BASIC'e geri döner.)

TAB (TABULATION: Tablolama) (Özel fonksiyon)

TAB (<sayısal>)

TAB fonksiyonunu takipçiyi, o anda bulunduğu satırın ilk karakterinden başlayarak <sayısal>'da verilmiş değer kadar ilerletir ve bu sayı kadar boşluk bırakır. <sayısal>'da verilmiş olan değer 0 ile 255 arasında olmak zorundadır. TAB fonksiyonu yalnızca PRINT yönergesi ile birlikte kullanılır. PRINT# yönergesi ile mantıksal dosya için kullanıldığında herhangi bir etkisi yoktur.

```
100 PRINT "ISIM" TAB(25) "MIKTAR": PRINT
110 NAM$="PATATES": AMT$="25"
120 PRINT NAM$ TAB(25) AMT$
RUN
ISIM                     MIKTAR

PATATES                  25

READY.
```

TAN (TANGENT: Tanjant) (Sayısal fonksiyon)

TAB (<sayısal>)

<sayısal>'da verilen değerin radyan cinsinden tanjantını, verir. Eğer TAN değeri taşmaya (overflow) neden olursa **?DIVISION BY ZERO** (sıfıra bölünme) hatası meydana gelir .

```
10 XX = .785398163: YY=TAN(XX):PRINT YY
RUN
 1

READY.
```

TIME **(TIME: Zaman) (Sayısal fonksiyon)**

TI

TI fonksiyonu "aralık zamanlayıcısı"'nı okur. Bu tip bir "saat", "anlık saat" olarak adlandırılır ve değeri sistem açıldığında 0'a eşitlenir. Teyp giriş/ çıkış sırasında, bu 1/60 saniyelik, "aralık zamanlayıcısı" işleyişi durdurulur. İşlem bitince kaldığı yerden devam eder.

```
10 PRINT TI/60 "ACILDIGINDAN BERI GECEN
SANIYE"
```

TIME$ **(TIME: Zaman) (Yazınsal dizi fonksiyon)**

TI$

TI$ zamanlayıcısı sisteminizin açıldığı andan o ana kadar geçen zamanı okur ve gerçek zaman saati gibi çalışır. Sistemin donanımında yer alan aralık zamanlayıcısı (ya da onluk saat) TI$, size 6 karakterlik saati, dakika ve saniyeleri bir formatla gösterir. TI$ zamanlayıcısına, aynı kol saatinizi ayarlarken yaptığınız gibi geçici bir başlama noktası verebilirsiniz. Teyp giriş/çıkış işleminden sonra TI$ değeri doğruluğunu yitirir.

```
1 TI$ = "000000": FOR J=1 TO 10000: NEXT
: PRINT TI$
RUN
000011

READY.
```

USR **(USER: Kullanıcı) (Sayısal fonksiyon)**

USR (<sayısal>)

USR fonksiyonu, başlangıç adresleri, belleğin 785-786'ncı yerlerinin içeriği olan, kullanıcı tarafından çağrılabilir makina dili alt-programlarına dallanmak için kullanılır. USR fonksiyonu kullanılmadan önce, başlangıç adresleri 785-786'ncı bellek yerlerine yerleştirilmelidir (POKE yönergesi kullanarak). Eğer POKE yönergesi kullanılmamış ise 785-786'ncı birimler **?ILLEGAL QUANTITY** (Kural dışı miktar) hatasına neden olurlar.

<sayısal>'ın değeri, assembler kod ile erişebilmek için 97'nci hafıza biriminden başlayan ondalık akümülatörde saklanmıştır ve USR fonksiyonunun sonucu alt-programdan BASIC'e dönüldüğünde yine burada yer alan değerdir. Yani bu komutun kullanımı, makina dili bilgisi gerektirir.

```
10 B=T * SIN(Y)
20 C=USR (B/2)
30 D=USR (B/3)
```

VAL (VALUE: Değer) (Sayısal fonksiyon)

VAL (<yazınsal-dizi>)

<yazınsal-dizi>'nin sayısal değerini verir. Eğer dizideki boşluk olmayan ilk karakter (+), (-) işareti ya da sayı değilse fonksiyonun değeri sıfırdır. Yazınsal dizi dönüşümü dizinin sonuna gelindiğinde ya da sayı olmayan bir karakter bulunduğunda sona erer. (Kesirli sayıların ifadesinde kullanılan nokta ve üs gösteren e harfi buna dahil değildir.)

```
10 NAM$="ISTANBUL" : ZIP$="34000"
20 IF VAL(ZIP$)>20000 OR VAL(ZIP$)<50000
THEN PRINT NAM$ TAB(25)"DAHA BUYUK ORDU"
RUN
ISTANBUL                DAHA BUYUK ORDU


READY.
```

VERIFY (VERIFY: Kontrol et) (Komut)

VERIFY ["<dosya-ismi>"] [,<cihaz>]

VERIFY komutu, doğrudan veya program modunda, kasetteki ya da disketteki BASIC program dosyasının içeriklerini, o anda bellekte var olanla kıyasılar. VERIFY genellikle SAVE işleminden hemen sonra, programın kaset veya diskette doğru olarak saklanıp saklanmadığını, kontrol etmek için kullanılır.

Eğer <cihaz> numarası belirtilmemişse, bilgisayar cihaz numarası 1 olan teybi seçtiğinizi varsayar. Kasetteki dosyalar için <dosya-ismi> verilmemişse kasette o anda gelen program ile bellekteki kıyaslanır. Disketteki (cihaz numarası 8) dosyalar için <dosya-ismi> verilmesi zorunludur. Eğer kıyaslama sonucunda program metinleri arasında farklılık varsa **?VERIFY ERROR** oluşur.

Programın adı, ("") tırnak içinde veya bir yazınsal dizi değişkeni gibi verilebilir. VERIFY aynı zamanda kasette yer alan son programın bitiş yerini saptamak için de kullanılabilir. Böylece yeni programın, kaza ile kasette yer alan programların üzerine saklanması önlenmiş olur.

```
VERIFY

PRESS PLAY ON TAPE
OK

SEARCHING
FOUND DOSYA-ISMI
VERIFYING
OK

READY.
```

(Kasetteki ilk programı kontrol eder.)

```
10 SAVE "DENEME",8
20 VERIFY "DENEME",8
```

(Diskete DENEME ismiyle programı kaydeder ve hemen ardından 8'inci cihazda programı kontrol eder)

WAIT (WAIT: Bekle) (Yönerge)

WAIT <konum>, <maske-1> [,<maske-2>]

Bu yönerge, verilen bir bellek adresi, belli bir bit örneğini tanıyıncaya kadar programın işletimini durdurmak amacıyla kullanılır. Diğer bir deyişle WAIT, belirli bir dış olay oluşuncaya kadar programı durdurmakta kullanılır. Bu, Giriş/Çıkış kayıtlarındaki durum bitleri kontrol edilerek yapılır. WAIT ile kullanılan veri parçaları, sayısal ifadelerden herhangi birisi olabilirler ama, bunlar tamsayı değerlerine çevrileceklerdir.

Birçok programcı bu yönergeyi asla kullanmaz. Bu, belirli bellek yerleşimindeki bitler, belli bir yoldan değiştirilinceye kadar programın durdurulmasına neden olur. Bu da, belli giriş/çıkış işlemleri için kullanılır, başka bir şey için değil.

WAIT yönergesi bellek yerleşimindeki değeri alır ve <maske-1>'deki değer ile mantıksal AND (ve) işlemine sokar. Yönergede <maske-2>'de verilmişse, ilk işlemin sonucu <maske-1> ile özel OR işlemine sokulur. Diğer bir deyişle <maske-1> test etmek istediğiniz birtakım bitleri ortadan kaldırır. <maske-1>'de 0 olan bit, sonuçtaki bitin her zaman 0 olmasını sağlar. 0 olup olmadığı kontrol edilen herhangi bir bit, <maske-2>'de karşılık gelen konumunda 1 olmalıdır.

Eğer <maske-1> ve <maske-2>'de karşılıklı bitler değişik şekilde işliyorlarsa, özel olan OR işlemi bit sonucunun 1 olmasına neden olur. Eğer karşılıklı bitlerden aynı sonuç elde edilirse bitin sonucu 0'dır. WAIT yönergesi ile sonsuz bir bekleyişe girmek mümkündür. **RUN/STOP** ve **RESTORE** tuşlarına aynı anda basın. Aşağıdaki ilk örnekte programa devam edebilmek için teyp ünitesinin üzerindeki play/geri-ileri sarma tuşuna basılması beklenmektedir. İkinci örnekte ise bir yaratık, zemin ekranına çarpıncaya kadar beklenecektir.

```
WAIT 1, 32, 32
```

```
WAIT 53273, 6, 6
```

```
WAIT 36868, 144, 16
```

(144 ve 16 maskeleridir. 144 binary olarak 10010000'e, 16 ise 10000'e eşittir. WAIT yönergesi 128'inci bit 1 ya da 16'ncı bit 0 olana kadar programın durmasını sağlar.)

COMMODORE 64'ÜN KLAVYESİ VE ÖZELLİKLERİ

İşletim sisteminin, klavyeden girilen karakterleri işleme sokuluncaya kadar tutan, 10 karakterlik bir klavye "deposu" (buffer) bulunur. Bu depo ya da kuyruk, karakterleri girildiği sırayla tutar ve ilk girilen karakterler ilk işlenir. Yani ilk basılan tuş, kuyruktaki ilk değerdir ve ilk olarak bu tuş işleme sokulur. Örneğin: eğer ilk basılan tuş işleme sokulmadan ikinci bir tuşa basılırsa, bu ikinci karakter, ilk basılan karakterin işlemi bitinceye kadar depoda saklanır. İlk karakter ile yapılacak işlemler sona erdikten sonra, depoda daha veri olup olmadığı kontrol edilir ve basılan ikinci tuş işleme sokulur. Eğer bu depo olmasaydı, klavyeden hızlı hızlı veri girişi yapıldığında, birtakım karakterler kaybolacaktı.

Diğer bir deyişle, klavye deposu sizin tuşlara arka arkaya basmaya izin verir. Tuşlara bastığınızda bu tuşların karakter değerleri, depo içindeki tek dosyada, basılış sırasıyla işlenmek için sıralarını beklerler. Arka arkaya basılabilme özelliği, programın, yanlışlıkla basılmış olan tuşların karakterlerini de depodan almasına neden olur.

Normalde, yanlışlıkla basılan tuşlar herhangi bir problem yaratmaz. Çünkü bu karakterleri takipçi kontrol tuşları (özellikle CuRSoR-Left **←CRSR** veya DELete **INST/DEL** tuşları) ile düzelterek doğru karakterleri girmeniz mümkün. Düzeltmeler **RETURN** tuşu algılanmadan önce işleme sokulacaklardır. Bununla birlikte, **RETURN** tuşuna basıldıktan sonra herhangi bir düzeltme mümkün değildir, çünkü, **RETURN** tuşu dahil tüm karakterler depolanıp herhangi bir düzeltmeden önce işleneceklerdir. Okuma işleminden önce klavye deposunu boşaltan bir döngü kullanarak bu durumdan kurtulabilirsiniz:

```
10 GET JUNK$: IF JUNK$ <>"" THEN 10: REM
KLAVYE DEPOSUNU BOSALT
```

GET ve INPUT'a ek olarak, PEEK komutu ile 197 ($00C5) adresindeki bellek yerleşiminden o anda basılmış olan tuşun tamsayı değerini elde etmek mümkündür. PEEK komutu işlenirken hiçbir tuşa basılmamışsa elde edilecek olan değer 64'tür. Sayısal klavye değerleri, klavye sembolleri ve karakter karşılıkları (CHR$), EK C'de verilmiştir. Aşağıda verilen örnek bir tuşa basılıncaya kadar bekler, sonra da tam sayıyı bir karakter değerine dönüştürür.

```
10 AA = PEEK(197): IF AA = 64 THEN 10
20 BB$ = CHR$(AA)
```

Klavye 8 satıra 8 sütunluk bir matris şeklinde düzenlenmiş bir seri anahtardan oluşmuştur. Tuşun basılı olup olmadığını anlamak için, klavye matrisi KERNAL tarafından CIA#1 giriş/çıkış çipi (MOS 6526 Complex Interface Adapter) kullanılarak taranır. Taramayı yapmak için iki CIA kaydı kullanılır. Bunlar; klavyenin sütunları için 56320($DC00) adresindeki 0 nolu kayıt ve klavyenin satırları için 56321 ($DC01) adresindeki 1 nolu kayıt.

56320 adresindeki 0-7'nci bitler, 0-7 sütunlarına, 56321 adresindeki 0-7'nci bitler, 0-7 satırlarına karşılık gelir. Sütun değerlerini sırasıyla yazıp sonra da satır değerlerini okuyarak, KERNAL kapanan anahtarlarının kodlarını, basılan tuşun CHR$(N) değerine yerleştirir.

Sekiz sütuna sekiz satır, 64 olası değer demektir. Bununla birlikte ilk olarak **RVS** ,**CTRL** ya da **C=** tuşlarına bastığınızda, ya da **SHIFT** tuşuna basıp ikinci bir karakter girdiğinizde, ek değer üretilir. Bunun nedeni, KENAL'ın bu tuşları ayrı ayrı kodlarına ayırması ve kontrol tuşlarından birine basıldığında "hatırlaması"'dır. Klavye taramasının sonucu 197 adresine yerleştirilir.

Karakterler, POKE yönergesi kullanılarak 631-640 adreslerindeki klavye deposuna doğrudan yazılabilirler. Bu karakterler. 198 adresindeki karakter sayacı POKE kullanılarak set edildiğinde (1 yapıldığında) işleme sokulacaklardır.

Bunlar, yönergelerin ekran üzerine basılarak, depoya RETURN işareti koyularak ve sonra karakter sayacı bir yapılarak, bir seri direkt-mod komutlarının işletilmesini sağlayacak şekilde kullanılabilirler. Aşağıdaki örnekte, program kendi listesini yazıcıdan dökecektir ve kaldığı yerden işlemeye devam edecektir.

```
10 PRINT CHR$(147) "PRINT#1: CLOSE 1: GO
TO 50"
20 POKE 631, 19: POKE 632, 13: POKE 633,
13: POKE 198, 3
30 OPEN 1, 4: CMD1: LIST
40 END
50 REM PROGRAM BURADAN TEKRAR BASLAYACAK
```

EKRAN EDİTÖRÜ

Ekran editörü, program metninin düzenlenmesi için kolaylıklar sağlar. Programın listesi ekranda yer aldığında takipçi kontrol tuşları ve diğer özel tuşları kullanarak ekranda istediğiniz yere gelip, istediğiniz değişikliği yapabilirsiniz. Program metninin, satır numarası belli olan bir satırında istediğiniz tüm değişiklikleri yaptıktan sonra satırın herhangi bir yerindeyken **RETURN** tuşuna basmakla, ekran editörünün, 80 karakterlik mantıksal ekran satırının tümünü okumasını sağlamış olursunuz.

Bundan sonra metin ayrıştırılıp işlenmek ve programın içinde yer almak için yorumlayıcıya (interpreter) devredilir. Düzeltilen satır, bellekteki eski satırla yer değiştirir. Herhangi bir satırın aynısının programda yer almasını istiyorsak, sadece bu satırın satır numarasını değiştirmek ve **RETURN** tuşuna basmak yeterlidir.

Editör ekrandaki iki fiziksel ekran satırını okuyabilir. Bu yüzden, eğer satırda kısaltılmış sözcükler kullandıysanız ve bunların açık yazılımları ile satırın uzunluğu 80 karakterden fazla oluyorsa, takipçi bu satırda iken **RETURN** tuşuna basarsanız, editör üçüncü satıra taşmış olan karakterleri okuyamaz. Bu nedenle INPUT yönergesi ile de, uzunluğu toplam 80 karakterden fazla olan verilerin okunması da mümkün değildir. Böylece, amaçlarınıza uygun BASIC metninin satır uzunluğu ekranda da görebileceğiniz gibi 80 karakterle sınırlıdır.

Belirli durumlarda ekran editörü takipçi kontrol tuşlarını, normalde işlediğinden farklı bir şekilde değerlendirir. Eğer bu tuşlar tırnak işaretinin hemen sağında kullanılmışsa editör bu tuşları tırnak işareti moduna uygun bir biçimde işler.

Tırnak işareti modunda veri karakterleri normalde olduğu gibi girilir. Takipçi kontrol karakterleri ise tersine bir biçimde farklı karakterler olarak görüntülenirler ve program işletildiğinde görevlerini yerine getirirler. Renk kontrol tuşları için de aynı şeyler geçerlidir. Bu size, takipçi ve renk kontrol tuşlarını, programınızdaki yazınsal dizi verileri içinde kullanma olanağı verir. Bunun çok önemli bir özellik olduğunun farkındasınızdır. Çünkü, tırnak işaretleri arasında yer alan metin ekranda görüntüleneceği zaman, takipçi konumlama ve renk kontrol fonksiyonları, yazınsal dizinin bir parçası şeklinde otomatik olarak işleme sokulurlar. Yazınsal dizi içinde takipçi kontrollerinin kullanılmasına bir örnek verirsek:

```
10 PRINT "A]]B|||C]]D"
RUN
AC BD

READY.
```

"A(Sağ)(Sağ)B(Sol)(Sol)(Sol)C(Sağ)(Sağ)D" (Sağ)=CRSR Sağa (Sol)=CRSR Sola

DEL tuşu, tırnak işareti modundan etkilenmeyen tek takipçi kontrol tuşudur. Bu yüzden tırnak işareti modunda iken herhangi bir hata yaptığınızda ←CRSR tuşu ile hatalı karakter üzerine gidip düzeltme yapılmadığından, DEL tuşunu kullanabilirsiniz. INST tuşu da tırnak işareti modunda iken tersine olarak görüntülenir. Tırnak işareti modunda çalışırken tüm satırı girmeden önce RETURN tuşuna basarak tuşları normal modda olduğu gibi kullanabilirsiniz. Diğer bir seçenek, eğer bu modda iken takipçi kontrol tuşlarını kullanmaya gerek duymuyorsanız. Tırnak işareti modundan kurtulmak için RUN/STOP ve RESTORE tuşlarına basmaktır. Tırnak işareti modunda iken yazınsal dizi içinde kullanabileceğiniz takipçi kontrol tuşları Tablo 2-2'de verilmiştir.

Tablo 2-2. Tırnak işareti modunda kullanılan takipçi kontrol karakterleri

AÇIKLAMA	KONTROL TUŞU	EKRAN GÖRÜNTÜSÜ
Takipçi yukarı	↑CRSR	
Takipçi aşağı	CRSR↓	
Takipçi sola	←CRSR	
Takipçi sağa	CRSR→	
Temizle	CLR/	
En üst sol köşe	/HOME	
Araya yerleştir	INST/DEL	

Tırnak işareti modunu kullanmıyorsanız SHIFT tuşuna basarken INST tuşuna da basarsanız takipçinin sağ tarafında kalan veriler sağa doru kayar ve karakterler arasında yeni girişler yapabilmeniz için boşluk açılır. Editör, açılan boşluk dolduruluncaya kadar INSERT modunda çalışmaya başlar.

INSERT modunda iken de takipçi ve renk kontrol karakterleri tersine olarak görüntülenirler. Tek fark, INST/DEL tuşunun araya sokulması (INSERT) ve aradan çıkartılması (DELETE) sırasında oluşur. Tırnak işareti modunda iken DEL tuşu normal işlevini yerine getirmesine rağmen, bu modda iken tersine bir T üretecektir. INST tuşu ise, tırnak işareti modunda iken tersine bir karakter üretmesine rağmen, bu modda normal işlevini sürdürecektir.

Bu PRINT yönergesi içinde, tırnak işareti modunda iken kullanmadığınız DEL tuşunu kullanabilmenizi sağlar. INSERT modundan, RETURN, SHIFT ve RETURN veya RUN/STOP ve RESTORE tuşlarına basarak kurtulabilirsiniz. Ya da bu moddan araya yerleştirilmiş (INSERT) tüm boşlukları doldurarak kurtulabilirsiniz. DEL karakterlerinin bir yazınsal dizi içindeki kullanımlarına ilişkin bir örnek verirsek:

10 PRINT "SULAK" DEL INST INST DEL DEL H"

Tuşlar yukarıda verilen sırada girilmelidir, bellekteki programın listesi alındığında görünümü aşağıdaki gibi olacaktır.

10 PRINT "SULH"

Uzman programcıların tercih edeceği bir yöntem değildir.

Bu örneği RUN yazıp işlettiğinizde SULH sözcüğü görüntülenecektir. Çünkü AK harfleri, H basılmadan önce çıkartılacaktır. Çıkarılan (DELete) karakter yazınsal dizi içinde PRINT yönergesi için olduğu kadar LIST yönergesi için de çalışacaktır. Bunu, metnin bir satırını ya da bir bölümünü saklamak için bir teknik oluşturmakta kullanabilirsiniz. Bununla birlikte bu karakterleri içeren bir satırı düzeltmeye çalışmak, imkânsız olmasa bile oldukça zordur.

Klavyeden girilmesi kolay olmayan, fakat özel amaçlar için basılması gereken bazı karakterler vardır. Bunları tırnak işaretleri içerisinde elde etmek yerine, satırda bunlar için boş yerler bırakmalı. RETURN tuşuna basmalı ve sonra satırı düzeltmek için geri dönmelisiniz. Şimdi tersine karakterler elde etmek için, parmağınız CTRL (Kontrol) tuşu üzerindeyken RVS/ON tuşuna basın. Tuşları aşağıda verildiği gibi

FONKSİYON	KONTROL TUŞU	EKRAN GÖRÜNTÜSÜ
SHIFT RETURN	SHIFT M	
Küçük harfe geçiş	N	N
Büyük harfe geçiş	SHIFT N	
Büyük/küçük harf değiştirme tuşlarını devre dışı bırakma	H	H
Büyük/küçük harf değiştirme tuşlarını	I	I

SHIFT tuşu ile birlikte RETURN tuşuna basmak yazınsal diziyi bitirmez ama ekran üzerinde satır-başı yapılmasına neden olur. Bu, PRINT yönergesi için olduğu kadar LIST yönergesi için de geçerlidir. Böylece eğer bu karakter kullanılmışsa bu satırı düzeltmek hemen hemen imkansızdır. Çıktı, CMD yönergesi ile yazıcıya gönderildiğinde tersine "N" karakteri, yazıcının büyük/küçük harf karakter setine geçmesine ve SHIFT N de yazıcının büyük harf/grafik karakter setine geçmesine neden olur.

Tersine video karakterleri, CTRL ve RVS tuşlarına aynı anda basılarak yazınsal dizilerin içinde yer alabilirler. Bu durumda tırnak işaretlerinin içerisinde tersine R harfi belirecektir. Bu, tüm karakterlerin tersine video (negatif bir fotoğraf gibi) şeklinde basılmasını sağlar. Bu tersinelikten kurtulmak için CTRL ve RVS/OFF tuşlarına aynı anda basmak yeterlidir. Sayısal veriler, önlerine CHR$(18) yazılarak tersine video şeklinde basılabilirler. CHR$(146) ya da RETURN tuşuna basarak tersine video çıkışından kurtulabilirsiniz.

BÖLÜM 3

GRAFİKLERİN PROGRAMLANMASI

- Grafiklere Genel Bakış
- Grafik Yerleşim Noktaları
- Standart Karakter Modu
- Programlanabilir Karakterler
- Çok-Renkli Grafikler
- Geliştirilmiş Zemin-Rengi Modu
- Bit Grafikleri
- Çok-Renkli Bit Haritalama Modu
- Düz Kaydırma
- Yaratıklar
- Diğer Grafik Özellikleri
- Yaratık Programlama

GRAFİKLERE GENEL BAKIŞ

Commodore 64'ünüzün grafik yeteneklerinin nedeni 6567 Video Interface Chip'idir (VIC-II çip). Bu çip, 40 sütun x 25 satırdan oluşan metin gösterimi, 320'ye 200 noktalık yüksek çözünürlüklü gösterimler ve oyun yazımını kolaylaştıran küçük, hareketli nesneler SPRITES (yaratıklar) gibi çeşitli grafik modlarının oluşturulmasını mümkün kılar. Ayrıca birçok değişik grafik modu aynı ekranda birleştirilebilir. Örneğin: ekranın üst kısmı yüksek çözünürlük modundayken, alt kısmı yazı modunda olabilir. Yaratıkları ise her türlü modla birleştirmeniz mümkündür. Biz öncelikle "yaratıklar" dışındaki grafik modları üzerinde duracağız. Yaratıklar için daha ayrıntılı bilgiler ise kitabın daha sonraki bölümlerinde yer alıyor.

VIC-II çipi aşağıdaki grafik gösterim modlarına sahiptir:

A. KARAKTER GÖSTERİM MODLARI:

1. Standard Karakter Modu
- a. ROM karakterleri
- b. RAM programlanabilir karakterleri

2. Çok-Renkli Karakter Modu
- a. ROM karakterleri
- b. RAM programlanabilir karakterleri

3. Genişletilmiş Arka Plan Renk Modu
- a. ROM karakterleri
- b. RAM programlanabilir karakterleri

B. BİT HARİTA MODLARI:

1. Standard Bit Harita Modu

2. Çok-Renkli Bit Harita Modu

C. YARATIKLAR:

1. Standard Yaratıklar

2. Çok-Renkli Yaratıklar

GRAFİKLERİN KONUMLARI

Önce bir takım genel bilgiler verelim. Commodore 64'ünüzün ekranında 1000 olası yerleşim noktası vardır. Normalde ekranınız 1024'üncü (onaltılık gösterimle $0400) noktadan başlar ve 2023'uncü noktada sona erer. Bu noktaların her birinin genişliği 8 bittir. Yani bir nokta 0'dan 255'e kadar olan herhangi bir tam sayıyı taşıyabilir. Ekran belleğine bağlı olarak 1000 birimlik bir bölge de RENK BELLEĞİ veya RENK RAM'ı (COLOR MEMORY veya COLOR RAM) olarak adlandırılır. Bu bölge, 55296 ($D800)'ıncı noktadan başlar ve 56295'te sona erer. Her renk noktası 4 bit genişliğindedir. Yani her renk noktası 0'dan 15'e kadar olan herhangi bir tam sayıyı içerebilir. Commodore 64 ile, 16 çeşit renk elde edebilmeniz, bu sistem sayesindedir.

Normal ekran gösteriminde, ekran belleğindeki 1000 yerleşim noktasının her biri bir kod numarası taşır. Kod numaraları VIC-II çipine, bu yerleşim noktasında hangi karakterin görüntüleneceğini bildirir. Commodore grafik modları, VIC-II çipinde yer alan 47 kontrol kaydı tarafından seçilir. Grafik fonksiyonları, gereken değerlerin, kayıtlarda birine yerleştirilmesiyle (POKE komutunu kullanarak), kontrol edilir. VIC-II çipinin kontrol kayıtları, belleğin 53248'den ($D000) 53294'e ($D02E) kadar olan bölümünde yer alırlar.

VİDEO KÜME SEÇİMİ

VIC-II çipi bir kerede belleğin sadece 16K'lık bölümüne erişebilir. Commodore 64'ünüzün 64K'lık belleği, 16K'lık 4 Kümeye (bölüme) bölünmüştür. VIC-II çipinin, bu bellek bölgelerinden herhangi birini görmesini sağlamak mümkündür. Belleğin farklı bölümlerine erişmenizi sağlayan KÜME-SEÇİM (BANK-SELECT) bitleri 6526 COMPLEX INTERFACE ADAPTER CHIP #2 (CIA #2) diye adlandırılan çipte yer alırlar. BASIC yönergelerinden olan PEEK ve POKE (veya bunların makina dilindeki biçimleri), CIA #2'nin A KANAL'ındaki 0 ve 1 bitlerini kontrol ederek (56576 ($DD00)), bir küme seçilmesini sağlarlar. Küme seçmek için 56578 ($DD02)'nci yerleşimin 0 ve 1'inci bitleri 1 yapılarak CIA #2, "çıkış" (output) için hazırlanmalıdırlar. Aşağıdaki örnek bu konuda yardımcı olacaktır:

```
POKE 56578, PEEK(56578) OR 3 :REM 0 VE 1
'INCI BITLERI 'CIKIS' ICIN HAZIRLA
```

```
POKE 56576, (PEEK(56576) AND 252) OR A:
REM KUME DEGISTIR
```

Yukarıdaki örnekte verilen “A” değeri aşağıdaki tabloda verilen değerlerden birini almak zorundadır.

A DEĞERİ	BİT	KÜME	BAŞLANGIÇ KONUMU	VIC-II ÇİP ARALIĞI
0	00	3	49152	($C000-$FFFF)*
1	01	2	32768	($8000-$BFFF)
2	10	1	16384	($4000-$7FFF)*
3	11	0	0	($0000-$3FFF)(VARSAYILAN DEĞER)

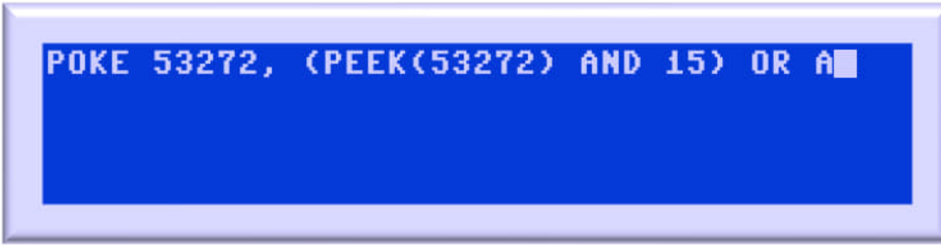
***NOT:** VIC-II, 1 ve 3'üncü kümelerde normal Commodore 64 karakter setine ulaşamaz. (Karakter belleği bölümüne bakınız.)

VIC-II çipinin yaptığı her şey, bu 16K'lık Küme kavramına dayalıdır. Daima VIC-II çipinin hangi kümeyi gösterdiği konusunda uyanık olmalısınız, çünkü bu, karakterleri tanımlayan görüntü verilerinin nereden geldiğini, ekranın nerede olduğunu, yaratıkların nereden geldiklerini vb. etkileyecektir. Commodore 64'ünüzü açtığınızda, 56576 adresindeki 0 ve 1'inci bitler otomatik olarak KÜME 0'a ($0000-$3FFF) ayarlanırlar.

EKRAN BELLEĞİ

Ekran belleğinin yeri, 53272 ($D018)'deki kontrol kaydına yapılacak bir POKE ile değiştirilebilir. Bu aynı zamanda, hangi karakter setinin kullanılacağını da belirlediğinden, kontrol kaydının karakter seti ile ilgili bölümünü bozmamaya dikkat etmelisiniz. ÜST 4 bit ekran belleğinin yerini denetler.

Ekranı başka yere taşımak isterseniz aşağıdaki yönergeyi kullanmalısınız:



Yukarıdaki örnekte verilen "A" değeri aşağıdaki tabloda verilen değerlerden birini almak zorundadır.

A DEĞERİ	BİT	KONUM*	
		ONLUK	ONALTILIK
0	0000XXXX	0	$0000
16	0001XXXX	1024	$0400 (Varsayılan)
32	0010XXXX	2048	$0800
48	0011XXXX	3072	$0C00
64	0100XXXX	4096	$1000
80	0101XXXX	5120	$1400
96	0110XXXX	6144	$1800
112	0111XXXX	7168	$1C00
128	1000XXXX	8192	$2000
144	1001XXXX	9216	$2400
160	1010XXXX	10240	$2800
176	1011XXXX	11264	$2C00
192	1100XXXX	12288	$3000
208	1101XXXX	13312	$3400
224	1110XXXX	14336	$3800
240	1111XXXX	15360	$3C00

***NOT:** VIC-II çipinin KÜME ADRESİNİ buradaki sayıya eklemeniz gerekir. Aynı zamanda KERNAL'ın ekran editörüne; POKE 648, sayfa ... komutunu kullanarak ekranın bulunduğu yeri söylemeniz gerekir. (Sayfa=adres/256, örneğin; 1024/256= 4, öyleyse POKE 648,4).

RENK BELLEĞİ

Renk belleğinin yeri değiştirilemez. Bu bellek daima 55296 ($D800) ve 56295 ($DBE7) adresleri arasındadır. Ekran belleği (1024'ten başlayan 1000 yerleşim) ve renk belleği farklı kullanımlara sahiptir. Bir grafik modunda yaratılan bir resim, bir başka grafik modunda tamamıyla farklı görünecektir.

KARAKTER BELLEĞİ

VIC-II'nin, ekranda görülen karakterleri tanımlayan bilgileri nereden aldığı, grafiklerin programlanmasında büyük önem taşır. Normalinde çip, görmek istediğiniz karakterin şeklini KARAKTER ROM (CHARACTER GENERATOR ROM)'dan alır. Karakter ROM'u ekrana çıkan harf, sayı, noktalama işaretleri gibi tüm karakter şekillerinin tanımlarını içerir. Commodore 64'ün bir özelliği de, Karakter ROM'u devre dışı bırakıp. RAM belleğinde yer alan karakter tanımlarını kullanabilmesidir. Bir başka deyimle, karakter tanımlarını kendiniz yaratıp RAM'e yerleştirebilirsiniz. Bu size, oyun ve iş uygulamalarınız için sonsuz sayıda sembol yapma şansı verecektir.

Normal bir karakter seti 256 karakterden oluşur ve her karakter 8 baytlık veriler ile tanımlanır. Her karakterin 8 bayt kapladığı düşünülürse tam bir karakter setinin 256 x 8 = 2K kadar bellek baytını kapladığını hesaplayabilirsiniz.

VIC-II çipi bir defada belleğin 16K'lık bir bölümüne erişebildiğine göre, tam bir karakter setini yerleştirmek için 8 farklı yer seçebilirsiniz. Şüphesiz, tam bir karakter setinden daha az karakter kullanabilirsiniz. Ancak yine de 8 başlama noktasının birinden başlamak zorundadır.

Karakter belleğinin yerini, 53272 ($D018)'deki VIC-II kontrol kaydının 3 biti belirler. Bit 3, 2 ve 1, 2K'lık bloklar halinde, karakter setinin nereye yerleştirildiğini belirler. Bit 0 göz önüne alınmaz. Aynı kayıtın, ekran belleğinin yerini belirlemek için de kullanıldığını hatırlayacaksınız. Aşağıdaki BASIC yönergesini karakter belleğinin yerini değiştirmek (bir yerden diğerine aktarmak) amacıyla kullanabilirsiniz:

```
POKE 53272, (PEEK(53272) AND 240) OR A
```

Yukarıdaki örnekte verilen “A” değeri aşağıdaki tabloda verilen değerlerden birini almak zorundadır.

A DEĞERİ	BİT	KONUM		
		ONLUK	ONALTILIK	
0	XXXX000X	0	$0000-$07FF	
2	XXXX001X	2048	$0800-$0FFF	
4	XXXX010X	4096	$1000-$17FF	0 ve 2’inci KÜME’lerde ROM’u görür (Varsayılan)
6	XXXX011X	6144	$1800-$1FFF	0 ve 2’inci KÜME’lerde ROM’u görür
8	XXXX100X	8192	$2000-$27FF	
10	XXXX101X	10240	$2800-$2FFF	
12	XXXX110X	12288	$3000-$37FF	
14	XXXX111X	14336	$3800-$3FFF	

NOT: KÜME adresini eklemeyi unutmayın.

Yukarıdaki tabloda ROM GÖRÜNTÜSÜ, karakter üreticisi ROM'a karşılık gelir. Küme 0'da yukarıdaki adresler için RAM'ın yerinde görünür. Ayrıca küme 2'deki 36864-40959 ($9000-$9FFF) adreslerinde de RAM'a karşılık gelir. VIC-II çipi bir defada sadece 16K'lık bir belleğe erişebildiğinden, ROM karakter kalıpları VIC-II çipinin baktığı belleğin 16K'lık blokunda yer alırlar. Bu yüzden, Commodore 64 sisteminin tasarımı, VIC-II çipi, veri kümesi 0 olduğunda ROM karakterlerinin 4096-8191 ($1000-$1FFF), 2 olduğunda ise 36864-40959 ($9000-$9FFF) yerleşimleri arasında olduğunu düşünecek (gerçekte karakter belleği 53248-57343 ($D000-$DFFF) arasındadır) şekilde yapılmıştır. Bu görüntüleme sadece VIC-II çipi tarafından görülen karakter verilerine uygulanmaktadır. Tıpkı diğer RAM belleği gibi, programlar için de kullanılabilir, bu tasarım biçimi kullanıcıyı etkilemez.

NOT: Eğer ROM görüntüleri, sizin kendi grafiklerinize karışıyorsa, KÜME SEÇİMİ BİTİNİ KÜME 1 ve KÜME 3 dışındaki bir KÜME'ye ayarlayın. Böylelikle ROM görüntülerinin, yolunuzun üzerine çıkmasını önlemiş olursunuz.

ROM'da karakter setinin adresi, yerleşimi ve kapsamı şöyledir:

BLOK	ADRES		VIC-II GÖRÜNTÜSÜ	AÇIKLAMA
	ONLUK	ONALTILIK		
0	53248	D000-D1FF	1000-11FF	Büyük harf karakter
	53760	D200-D3FF	1200-13FF	Grafik karakter
	54272	D400-D5FF	1400-15FF	Tersine büyük harf karakter
	54784	D600-D7FF	1600-17FF	Tersine grafik karakter
1	55296	D800-D9FF	1800-19FF	Küçük harf karakter
	55808	DA00-DBFF	1A00-1BFF	Büyük harf ve grafik karakter
	56320	DC00-DSFF	1C00-1DFF	Tersine küçük harf karakter
	56832	DE00-DFFF	1E00-1FFF	Tersine büyük harf ve grafik karakter

Keskin gözlü okuyucularımız hemen şuna dikkat edeceklerdir. Karakter ROM'unun kapladığı yer ile VIC-II çipinin kontrol kayıtlarının yerleri aynıdır. Bunun nedeni aynı yerlerin "aynı zamanda" kullanılmamasıdır. VIC-II, bir karakter verisine erişmek istediğinde ROM'un kullanılabilmesi gerekir. ROM belleğin 16K kümesinde, VIC-II çipinin aradığı bir görüntü oluşur. Yoksa alan, Giriş/Çıkış kontrol kayıtları tarafından işgal edilir ve karakter ROM'una sadece VIC-II çipi erişebilir.

Ancak bu arada, programlanabilir karakterleri kullanıyorsanız, karakter ROM'unun bazı bölümlerinin, karakter tanımlarınızda kullanmak için, aktarılmasını isteyebilirsiniz. Bu durumda Giriş/Çıkış kayıtlarını aradan çıkarın, karakter ROM'unu işler duruma getirin ve karakter tanımlarınızda kullanacağınız aktarmaları yapın. Bu işlem bitince Giriş/Çıkış kayıtlarını tekrar eski durumuna getirmek zorundasınız. Aktarma süresince (Giriş/Çıkış devreden çıkarıldığında) hiçbir kesintiye izin verilmez. Çünkü kesintileri değerlendirmek Giriş/Çıkış kayıtlarının görevidir. Eğer bir kesintiye neden olursanız, gerçekten de garip şeylerin olduğunu göreceksiniz. Aktarma sırasında, klavyenin okunması mümkün değildir. Klavyeyi ve diğer normal kesintileri aradan çıkartmak için aşağıdaki POKE yönergesini kullanmalısınız:

```
POKE 56334, PEEK(56334) AND 254 : REM (K
ESINTILERI KAPATIR)
```

ROM'dan karakterlerin alınması işlemi bittikten sonra programınıza devam etmek istiyorsanız aşağıdaki yönergeyi kullanın:

```
POKE 56334, PEEK(56334) OR 1 : REM (KESI
NTILERI ACAR)
```

Aşağıdaki POKE yönergesi giriş/çıkışı aradan çıkarıp karakter ROM'unun kullanılmasını sağlar.

```
POKE 1, PEEK(1) AND 251
```

Artık karakter ROM'u 53248-57343 ($D000-$DFFF) adresleri arasındadır.

Normal işlemler için giriş/çıkışı, $D000'a geri almak isterseniz şu POKE yönergesini kullanın:

```
POKE 1, PEEK(1) OR 4
```

STANDART KARAKTER MODU

Commodore 64'unüzü açtığınızda bilgisayarınızın içinde bulunduğu mod, standart karakter modudur. Programlarınız çoğunlukla bu modda çalışır.

Karakterler ROM'dan veya RAM'dan alınabilir, ancak çoğunlukla kullanılan ROM'dur. Programınızda özel grafik karakterleri kullanmak istiyorsanız, bütün yapacağınız RAM'da yeni karakter biçimleri tanımlamak ve VIC-II'nin karakter bilgilerini ROM'dan değil RAM'dan almasını sağlamaktır. Bunları bir sonraki bölümde daha ayrıntılı olarak açıklayacağız.

Karakterlerin ekranda renkli olarak görünmesi için VIC-II, ekran belleğine erişir ve oradan ekranın söz konusu yerine gelecek karakter kodunu belirler.

Aynı zamanda VIC-II, renk belleğine de erişerek karakter için istediğiniz rengin tanımlanmasını sağlar. VIC-II, karakter kodunu, karakter biçiminizin bulunduğu 8 baytlık blokun başlama adresine çevirir. 8 baytlık blok karakter belleğinde yer alır.

Çevirme işlemi çok karışık değildir. Önce ekran belleğine POKE komutu ile yerleştirdiğiniz karakter kodu 8'le çarpılır. Buna karakter belleğinin (KARAKTER BELLEĞİ Bölümüne bakın) başlangıcı eklenir. Daha sonra ise bunlara temel adres eklenerek Küme Seçim Bitleri de hesaba alınmış olur. Tüm işlemleri şöyle formüle edebiliriz:

KARAKTER ADRES=EKRAN KODU*8+(KARAKTER SETİ*2048)+(KÜME*16384)

KARAKTER TANIMLARI

Her karakter 8'e 8'lik nokta kümelerinin yanıp sönen noktaları tarafından biçimlenir. Commodore 64 karakter görüntüleri karakter üretici ROM çipinde, 8 baytlık setler halinde saklanırlar. Her bayt, karakterdeki bir satırın nokta biçimini temsil ederken, her bit bir noktayı temsil eder. Bit 0 ise, noktanın sönük olduğu, bit 1 ise noktanın yandığı anlaşılır.

ROM'daki karakter belleği, 53248'nci adresten başlar (giriş/çıkış devreden çıkarıldığı zaman). 53248 ($D000)'den 53255'e kadar olan ilk 8 bayt @ işaretinin biçimini içerir. Bunun ekran belleğindeki karakter kodu değeri ise, sıfırdır. Bundan sonra gelen 8 bayt ise (53256 ($D008)'den 53263 ($D00F)'e) A harfinin oluşturacak bilgiyi taşırlar.

GÖRÜNTÜ	İKİLİ	PEEK KODU
**	00011000	24
****	00111100	60
** **	01100110	102
******	01111110	126
** **	01100110	102
** **	01100110	102
** **	01100110	102
	00000000	0

Tüm bir karakter setinin her biri bellekte 2K (2048 bitlik) bir yer kaplar. 256 karakter vardır ve her karakter için 8 bayt kullanılır (256*8=2048). Birisi grafikler ve büyük harfler için, diğeri ise büyük ve küçük harfler için olmak üzere iki karakter seti olduğundan, ROM (karakter üreticisi), toplam 4K'lık yer kaplar.

PROGRAMLANABİLİR KARAKTERLER

Karakterler ROM'da saklandıklarından, belli bir ülkenin diline özgü karakterleri (ğ,ç gibi) oluşturmanın yolu yok gibi görünür. Oysa, VIC-II çipine karakterlerin nerede bulunacağını bildiren bellek yerleşimi, programlanabilir bir kaydıdır ve belleğin farklı bölümlerini göstermek üzere değiştirilebilir. Karakter belleği göstergecini, RAM'i işaret edecek şekilde değiştirerek, istediğiniz karakter setini oluşturabilirsiniz.

Oluşturduğunuz karakter setini RAM'e yerleştirirken dikkat etmeniz gereken çok önemli birkaç kural vardır. Bu kurallara ek olarak, aşağıdaki iki kuralı da göz önünde bulundurmalısınız:

1-Eğer VIC-II çipine karakterlerle ilgili bilgileri, RAM'da sizin hazırladığınız alandan almasını söyleyerek kendi karakter setinizi kullanıyorsanız, standart Commodore 64 karakterlerinizi kullanamazsınız. Bu sorunu çözmenin tek yolu, kullanmak istediğiniz harf, sayı veya standart Commodore 64 grafiklerini RAM'da bulunan kendi karakter belleğinize aktarmanızdır. Burada, düzenlemeyi istediğiniz gibi yapabilirsiniz.

2-Karakter setiniz bellekte BASIC programınızdan ayrı olarak bir yer tutar. Ancak bir BASIC programı için kullanabileceğiniz 38K yeterli olacağından, uygulamaların çoğunda hiçbir sorun çıkmaz.

NOT: BASIC programlarınız da RAM'de yer alacağından, oluşturduğunuz karakter setinin üzerine gelmemesine dikkat etmelisiniz.

Commodore 64'te kendi karakter setinizin başlayabileceği, ancak mecburen kullanılmaması gereken 2 yerleşim vardır: 0'ıncı ve 2048'inci yerleşimler. 0'ın kullanılmamasının nedeni: sistemin önemli verileri 0'ıncı sayfada saklamasıdır. 2048 ise; BASIC programınızın başlama noktasıdır. Ancak bunun dışında kendi karakter setiniz için ayrılmış 6 başlama noktası daha vardır.

BASIC'de kullanılmak üzere oluşturduğunuz karakter setinizi yerleştireceğiniz en iyi yer, 12288 ($3000) adresinden başlar. Bunu yapmanın yolu 53272 adresinin 4 alt birine 12 yerleştirmektir. Bunu şöyle bir yönergeyle yapabilirsiniz:

```
POKE 53272,(PEEK(53272) AND 240) + 12
```

Aniden ekranda bulunan tüm harfler işe yaramaz hale dönüşürler. Bunun nedeni, şimdiye kadar 12288'de rastgele baytlar dışında hiçbir karakterin bulunmamasıdır. Commodore 64'ünüzü önce **RUN/STOP** daha sonra da **RESTORE** tuşlarına basarak normale döndürebilirsiniz.

Artık grafik karakterler yaratmaya başlayabiliriz. Karakter setinizi BASIC'den korumak için BASIC programınızın kullanmasını önlediğiniz bellek yerlerini kullanabilirsiniz. Şunu yazın:

```
PRINT FRE(0)-(SGN(FRE(0))<0)*65535
```

Gördüğünüz sayı şu anda bellekte ne kadar boş yer olduğunu belirtir.

```
POKE 52,48: POKE 56,48: CLR
```

Şimdi şunu yazın:

```
PRINT FRE(0)-(SGN(FRE(0))<0)*65535
```

Değişikliği fark ettiniz mi? Şimdi BASIC'ın kullanabileceği bellek daha azdır. Artık BASIC'ten çaldığınız belleğe karakter setinizi yerleştirebilirsiniz.

İkinci basamak, karakterlerinizi RAM'a yerleştirmektir. Başlangıçta 12288 ($3000)'de, rastgele bir veri başlangıcı yer alır. Karakter biçimlerinizi, VIC-II'nin kullanması için RAM'a yerleştirebilirsiniz.

Aşağıda verdiğimiz program ROM'da bulunan 64 karakteri, kendi karakter setinize aktarmanızı sağlar.

```
5 PRINT CHR$(142): REM BUYUK HARFE GECIS
10 POKE 52, 48: POKE 56, 48: CLR: REM KA
RAKTERLER ICIN BELLEKTE YER AYIRMA
20 POKE 56334, PEEK (56334) AND 254: REM
 TUS TARAMA KESME ZAMANLAYICISINI KAPAT
30 POKE 1, PEEK(1) AND 251: REM KARAKTER
I DEGISTIR
40 FOR I= 0 TO 511: POKE I + 12288, PEEK
 (I + 53248) : NEXT
50 POKE1, PEEK(1) OR 4 :REM G/C'I DEGIST
IR
60 POKE 56334, PEEK(56334) OR 1: REM TUS
 TARAMA KESME ZAMANLAYICISINI BASLAT
70 END
```

Şimdi 53272'inci adrese, POKE komutu ile (PEEK (53272) AND 240) + 12 yerleştirin. Hiçbir şey değişmedi, değil mi? Artık Commodore 64'ünüzün karakter bilgilerini aldığı yer ROM değil RAM. Fakat karakterleri ROM'dan aktardığınız için, hiçbir farklılık göremezsiniz.

Şimdi karakteri kolaylıkla değiştirebilirsiniz. Ekranı silin ve @ yazın. Takipçiyi iki satır aşağı indirin ve şunu yazın:

```
FOR I = 12288 TO 12288 +7: POKE I, 255 -
 PEEK(I): NEXT
```

İşte tersine bir @ işareti oluşturdunuz!

NOT: Tersine karakterler, karakter belleğinde bit biçimleri tersine çevrilmiş karakterlerdir.

Şimdi takipçiyi tekrar programın üstüne çıkarın ve **RETURN** tuşuna basın. Böylece, ters çevrilmiş karakterleri normal durumuna getirmiş oldunuz. Ekran gösterim kodları tablosundan bakarak her karakterin RAM'da nerede bulunduğunu çıkarabilirsiniz. Hatırlamanız gereken tek şey karakterin 8 baytta (bellek hanesinde) saklandığıdır. Şimdi başlangıç için birkaç örnek verelim:

KARAKTER	EKRAN KODU	RAM'DEKİ BAŞLANGIÇ KONUMU
@	0	12288
A	1	12296
!	33	12552
>	62	12784

Hatırlarsanız biz sadece ilk 64 karakteri almıştık. Diğer karakterlerden birini elde etmek isterseniz yapmanız gerekenler değişir.

Diyelim ki istediğiniz, karakter numarası 154 olan tersinelenmiş bir Z olsun. Bunu Z'yi tersineleştirerek kendiniz yapabilirsiniz ya da ROM'da bulunan tersine karakter setini olduğu gibi aktarabilirsiniz veya ROM'da artık gerek duymadığınız karakterle değiştirebilirsiniz.

Şimdi artık > işaretine gereksinimimizin kalmadığını varsayalım > işaretini negatif Z ile değiştirmek için şunu yazın:

```
FOR I=0 TO 7: POKE 12784 + I, 255 - PEEK
 (I+12496): NEXT
```

Şimdi > tuşuna basın. Karşınızda tersine bir Z bulacaksınız. > tuşuna ne kadar basarsanız basın artık görünen tersine bir Z'dir. (Bu değişiklik gerçekten bir yanılsamadır. > tuşu ne kadar tersine Z olarak görünürse görünsün programınızdaki işlevi yine aynı olacaktır. > tuşunu kullanmanız gereken bir şeyi deneyin. Garip görünüyor, ama hala çalışıyor değil mi?)

Tekrarlayalım: Karakterleri ROM'a aktarabilirsiniz, hatta bu sadece bir karakter bile olabilir. Programlanabilir karakterleri, yani kendi karakterlerinizi oluşturabilmeniz için gereken tek bir aşama (en iyi aşama) kaldı.

Karakterlerin ROM'da nasıl saklandığını hatırlıyor musunuz? Her karakterin 8 baytlık gruplarda saklandığını ve baytların, bit biçimlerinin karakterleri doğrudan kontrol edebildiğini anlatmıştık. 8 baytı, biri diğerinin üzerine gelecek şekilde düzenleyerek her baytı 8 ikili sayı olarak yazarsanız, 8'e 8'lik bir matris elde edersiniz. Bu, karakterlerin görüntüsüne benzeyecektir. Bit 1 ise; o hanede bir nokta olacaktır, yok eğer bit 0 ise, o hane boş kalacaktır.

Kendi özel karakterlerinizi oluştururken bellekte aynı bu tip bir tablo oluşturursunuz. Şimdi önce NEW yazın ve şu yönergeleri girin:

```
10 FOR I = 12448 TO 12455: READ A: POKE
I, A: NEXT
20 DATA 60, 66, 165, 129, 165, 153, 66,
60
```

Şimdi programınızı işletin (RUN ile). Program T harfini gülen bir yüz karakteriyle değiştirecektir. Yüzü görmek için birkaç kez T yazın. 20 numaralı DATA yönergesindeki her sayı gülen yüz karakterinde bir sıradır. Yüzü ortaya çıkaran matris şöyle bir matristir:

	7	6	5	4	3	2	1	0	ONLU	İKİLİ
SATIR 0			*	*	*	*			00111100	60
1		*					*		01000010	66
2	*		*			*		*	10100101	165
3	*							*	10000001	129
4	*		*			*		*	10100101	165
5	*			*	*			*	10011001	153
6		*					*		01000010	66
SATIR 7			*	*	*	*			00111100	60

	7	6	5	4	3	2	1	0
0								
1								
2								
3								
4								
5								
6								
7								

ŞEKİL 3-1. Programlanabilir Karakter Levhası

Kendi karakterinizi yaparken Programlanabilir Karakter Levhası (şekil 3-1) size yardımcı olabilir. Bu 8 satır ve 8 sütundan oluşan bir matristir. Her satır ve sütun numaralanmıştır. [Her satıra bir ikili kelime olarak bakarsanız, numaralar, o bit pozisyonunun değeridir. Her biri 2'nin kuvvetleridir. En soldaki bit 128'e (2 üzeri 7), ondan sonra gelen 64'e (2 üzeri 6) eşittir. Bu 2 üzeri 0, yani 1'e eşit olan en sağ bite kadar böylece gider (bit 0)].

Matriste, noktanın yer almasını istediğiniz her yerleşime X yerleştirin. Karakteriniz hazırsa artık DATA yönergenizi oluşturabilirsiniz.

İlk satırla başlayın. X'i koyduğunuz her hanenin sütun numarasını, yukarıda açıkladığımız gibi 2'nin kuvveti olarak kullanın ve bu sayıyı bir yere kaydedin. İlk satırın tüm sütunları için bu işlemi yapın ve sonuçları toplayın. Bu sayı, bu satırı oluşturmak üzere DATA yönergesine koyacağınız sayıdır.

Aynı işlemleri şimdi de diğer satırlar için yapın (1-7). İşlemleri bitirdiğinizde, elinizde 0'la 255 arasında 8 sayı olmalıdır. Eğer sayılarınızdan biri, bu aralığın dışındaysa toplamlarını yeniden kontrol edin. Elde ettiğiniz sayılar bu aralıkta (0-255) olmak zorundadır. Eğer elinizde bulunan sayılar 8 taneden az ise bir satırı atlamışsınız demektir. Sayılarınızın 0 olması, yanlış olması anlamına gelmez. Sıfır satırları da diğer satırlar kadar önemlidir.

Yukarıda verdiğimiz örneğin 20 numaralı satırındaki sayıların yerine hesapladıklarınızı koyun ve programı RUN komutu ile çalıştırın. Daha sonra T tuşuna basın. T tuşuna her basışınızda yaptığınız karakteri göreceksiniz.

Eğer yaptığınız karakterin bu halinden hoşlanmadıysanız, DATA yönergesinde kullandığınız sayıları değiştirin ve programı karakteriniz size mutluluk verinceye kadar çalıştırın.

Yapacağınız bütün iş bu!

NOT: Sonucun iyi olmasını, istiyorsanız, karakterinizi oluşturan dikey çizgilerin en az 2 nokta (bit) genişliğinde olmasına dikkat edin. Bu, karakterinizin renginin TV ekranında renk bozulmasına uğramasını önler.

Commodore 64 karakter setindeki tüm karakterleri bu yöntem ile yeniden oluşturabilirsiniz. Bu yeni oluşum size tanınan sınırlar dahilinde oluşur. Commodore 64'ün karakter belleği en fazla 255 ise siz 256'ıncı karakteri tanımlayamazsınız. Bir karakter 8*8 matristen oluşması gerekirken siz 5*5, 8*9 veya 9*9 gibi bir matris kullanamazsınız.

Oluşturduğumuz karakterler 8*8 matrislerden oluştuğundan bunun anlamı 8*8*8*8*8*8*8*8=16.777.216 adet farklı görünümde karakter demektir. 1 karakter de 8*8=64 adet noktadan oluşacak demektir.

Aşağıda standart programlanabilir karakterlerin kullanıldığı bir program örneği veriyoruz:

```
10 REM * ORNEK 1 *
20 REM PROGRAMLANABILIR KARAKTERLER
31 POKE56334,PEEK(56334)AND254:POKE1,PEE
K(1)AND251: REM KLAVYE G/C'I KAPAT
35 FOR I= 0 TO 63: REM ROM'DAN KOPYALANM
ASI GEREKEN KARAKTER SINIRLARI
36 FOR J= 0 TO 7: REM HER KARAKTER ICIN
8 BAYT KOPYALA
37 POKE 12288+I*8+J,PEEK(53248+I*8+J): R
EM BIR BAYT KOPYALA
38 NEXT J,I: REM BIR SONRAKI BAYT YADA K
ARAKTERE GIT
39 POKE1, PEEK(1)OR4:POKE56334,PEEK(5633
4)OR1: REM KLAVYE G/C'I AC
40 POKE53272, (PEEK(53272)AND240)+12: RE
M KARAKTER GOSTERGECINI 12288'E AYARLA
60 FORCHAR=60TO63: REM 60'DAN 63'E KADAR
OLAN KARAKTERLERIN PROGRAMI
80 FORBYTE=0TO7: REM BIR KARAKTERIN TUM
8 BAYTINI YAP
100 READNUMBER: REM KARAKTER VERISININ 1
/8'NI OKU VE BELLEGE SAKLA
120 POKE12288+(8*CHAR)+BYTE,NUMBER
140 NEXTBYTE:NEXTCHAR: REM BIR SONRAKI B
AYT YADA KARAKTER
150 PRINTCHR$(147)TAB(255)CHR$(60);: REM
YAPILAN KARAKTERI EKRANA YAZ
155 PRINTCHR$(61)TAB(55)CHR$(62)CHR$(63)
170 GETA$: REM BIR TUSA BASILMASINI BEKL
ER
180 IFA$="" THEN 170
190 POKE53272,21: REM NORMAL KARAKTERLER
E GEC
195 REM KARAKTERLER ICIN GEREKEN VERILER
200 DATA 4,6,7,5,7,7,3,3
210 DATA 32,96,224,160,224,224,192,192
220 DATA 7,7,7,31,31,95,143,127
230 DATA 224,224,224,248,248,248,240,224
240 END
```

ÇOK-RENKLİ GRAFİKLER

Standart yüksek çözünürlük grafikler size, ekran üzerindeki en küçük noktaları bile kontrol etme olanağı sağlar. Noktaların karakter belleğindeki değerleri ya 0, ya da 1'dir. 1, bu noktanın kullanıldığı (yandığı), 0 ise kullanılmadığı (yanmadığı) anlamına gelir. Eğer nokta yanıyorsa, noktanın, sizin bu ekran noktası için seçtiğiniz rengi aldığını göreceksiniz. Eğer standart yüksek çözünürlük grafiklerle çalışıyorsanız. 8*8'lik karakterleri oluşturan noktaların ya arka plan, ya da ön plan renklerini alabildiklerini belirtelim. Bu özellik kimi zaman renk çözünürlüğü için sınırlayıcı bir etken olabilir. Örneğin; farklı renklere sahip iki çizginin çakışması bir soruna yol açabilir.

ÇOK-RENKLİ MOD BİTİ

Çok-renkli karakter moduna geçmek istiyorsanız aşağıda verdiğimiz POKE yönergesini kullanacaksınız. Bu yönergeyi kullanarak 53270 ($D016) adresindeki VIC-II kontrol kaydının 4'üncü bitine 1 yerleştirin.

```
POKE 53270, PEEK(53270) OR 16
```

Çok-renkli karakter modundan kurtulmak için ise aynı hanenin 4'üncü bitine 0 yerleştirmek zorundasınız. Bunu ise aşağıdaki POKE yönergesiyle yapabilirsiniz:

```
POKE 53270, PEEK(53270) AND 239
```

Çok-renkli grafikler, yüksek çözünürlük (high-resolution) grafiklerle karışabilir. Bu, renk belleğinde bulunan 3'üncü bit tarafından kontrol edilebilir. Renk belleği 55296 ($D800 HEX) adresinden başlar. Eğer renk belleğindeki sayı 8'den küçükse (0-7) video ekranında buna karşılık gelen yer, standart yüksek çözünürlük (0-7) arasında seçtiğiniz rengi alacaktır. Eğer renk belleğine yerleştirdiğiniz sayı 8'e eşit ya da 8' den büyükse (8-15), o zaman o bölge çok renkli modda görünecektir.

Renk belleğine bir sayı yerleştirerek (POKE komutu ile) ekranın belirli bir yerinde bulunan bir karakterin rengini değiştirmeniz mümkündür. 0'la 7 arasındaki her sayı, normal karakter renklerini verecektir. 8'le 15 arasındaki sayılar ise çok-renkli moda geçmenizi sağlar. Başka bir deyişle renk belleğindeki 3'üncü bite 1 yüklemeniz sizi çok-renkli moda geçirecektir. Renk belleğinin 3'üncü bitine 0 yüklemeniz ise yeniden yüksek çözünürlük moda geçmenizi sağlar.

Çok-renkli moda geçince: bir karakterde yer alan bitler hangi noktalar için hangi renklerin görüntüleneceğini saptarlar. Örneğin A harfini ele alalım. Aşağıda A harfinin resmini ve bit biçimini görüyorsunuz.

GÖRÜNTÜ	BİT ŞABLONU
`   **   `	00011000
`  ****  `	00111100
` **  ** `	01100110
` ****** `	01111110
` **  ** `	01100110
` **  ** `	01100110
` **  ** `	01100110
	00000000

Normal ya da yüksek çözünürlük modda bitin 0 olduğu yerlerde ekran rengi, 1 olduğu yerlerde ise karakter rengi görüntülenir. Çok-renkli mod bitleri, çiftler halinde kullanılır. Bunu şöyle gösterebiliriz:

GÖRÜNTÜ	BİT ŞABLONU
AABB	00 01 10 00
CCCC	00 11 11 00
AABBAABB	01 10 01 10
AACCCCBB	01 11 11 10
AABBAABB	01 10 01 10
AABBAABB	01 10 01 10
AABBAABB	01 10 01 10
	00 00 00 00

Yukarıda gösterilen görüntü bölgesinde AA olarak işaretlenmiş bölümler 1 numaralı zemin rengi ile çizilirler, CC olarak işaretlenmiş bölümler ise karakter rengini alırlar. Bit çiftleri; aşağıdaki çizelgede gösterilen kurallara göre bunu belirler.

BİT ÇİFTİ	RENK KAYDI	KONUM
00	0 No.lu zemin rengi (ekran rengi)	53281 ($D021)
01	1 No.lu zemin rengi	53282 ($D022)
10	2 No.lu zemin rengi	53283 ($D023)
11	Renk belleğinde en alt 3 Bit tarafından belirlenen renk	Renk RAM'ı

NOT: Yaratık ön plan rengi 10'dur. Karakter ön plan rengi 11'dir.

NEW yazdıktan sonra, şu örnek programı girin:

```
100 POKE53281,1:REM ZEMIN RENGI 0'I BEYA
Z YAP
110 POKE53282,3:REM ZEMIN RENGI 1'I TURK
UVAZ YAP
120 POKE53283,8:REM ZEMIN RENGI 2'I TURU
NCU YAP
130 POKE53270,PEEK(53270)OR16:REM COK-RE
NKLI MODA GEC
140 C=13*4096+8*256:REM C'YI RENK BELLEG
INI GOSTERECEK SEKILDE AYARLA
150 PRINTCHR$(147)"AAAAAAAAAA"
160 FORL=0TO9
170 POKEC+L,8:REM SIYAH TONLARINI KULLAN
180 NEXT
```

Ekran rengini beyaz, karakter rengini siyah, bir renk kaydını camgöbeği, diğerini ise turuncu olarak elde edeceksiniz.

Tabi ki karakter renklerini, boşluklara karakter kodlarını koyarak elde etmiyorsunuz, aslında yaptığınız: renklerle ilgili kayıtları kullanmak. Bu bellekte fazla yer kaplamanızı önler, çünkü 2 bit 16 rengi (zemin) veya 8 rengi (karakter) kullanmanızı sağlayabilir. Bu aynı zamanda basit hileler yapmanızı da mümkün kılacaktır. Örneğin, sadece dolaylı kayıtlardan birini değiştirmekle, o renkte çizilmiş tüm noktaları değiştirebilirsiniz. Böylece bir anda ekrana çizilmiş olan her şey ve zemin renkleri değişebilir. Aşağıdaki program, 1 No.lu zemin renk kaydını değiştirmek için kullanılabilir:

```
100 POKE53270,PEEK(53270)OR16: REM COK-R
ENKLI MODA GEC
110 PRINTCHR$(147)CHR$(18);
120 PRINT"■";:REM TURUNCU RENK ICIN C=+1
'E BASIN
130 FORL=1TO22:PRINTCHR$(65);:NEXT
135 FORT=1TO500:NEXT
140 PRINT"■";:REM MAVI RENK ICIN CTRL+7'
YE BASIN
145 FORT=1TO500:NEXT
150 PRINT"■BIR TUSA BASIN":REM SIYAH REN
K ICIN CTRL+1'E BASIN
160 GETA$:IFA$=""THEN160
170 X=INT(RND(1)*16)
180 POKE53282,X
190 GOTO160
```

Tırnak işareti modundaki C= ve CTRL renk tuşlarını kullanarak, karakteri istediğiniz renge ayarlayabilirsiniz.

Örneğin, şu komutu yazın:

```
POKE53270, PEEK(53270)OR16:PRINT"■";:REM
 CTRL+3 KIRMIZI/COK-RENKLI KIRMIZI

READY.
```

READY. sözcüğü ve diğer yazdıklarınız ekranda çok-renkli modda ve kırmızı renkte görüntülenecektir.

Aşağıda çok-renkli programlanabilir karakterlerle ilgili bir program örneği veriyoruz:

```
10 REM * ORNEK 2 *
20 REM PROGRAMLANABILIR COK-RENKLI KARAK
TERLERIN YAPILMASI
31 POKE56334,PEEK(56334)AND254:POKE1,PEE
K(1)AND251:REM KLAVYE G/C'I KAPAT
35 FORI=0TO63:REM ROM'DAN KOPYALANMASI G
EREKEN KARAKTER SINIRLARI
36 FORJ=0TO7:REM HER KARAKTER ICIN 8 BAY
T KOPYALA
37 POKE12288+I*8+J,PEEK(53248+I*8+J):REM
 BIR BAYT KOPYALA
38 NEXTJ,I:REM BIR SONRAKI BAYTA YADA KA
RAKTERE GIT
39 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334
)OR1:REM KLAVYE G/C'I AC
40 POKE53272,(PEEK(53272)AND240)+12:REM
KLAVYE GOSTERGECINI 12288'E AYARLA
50 POKE53270,PEEK(53270)OR16
51 POKE53281,0:REM ZEMIN RENGI 0 SIYAH
52 POKE53282,2:REM ZEMIN RENGI 1 KIRMIZI
53 POKE53283,7:REM ZEMIN RENGI 2 SARI
60 FORCHAR=60TO63:REM 60'DAN 63'E KADAR
OLAN KARAKTERKERIN PROGRAMI
80 FORBYTE=0TO7:REM BIR KARAKTERIN TUM 8
 BAYTINI YAP
100 READNUMBER:REM KARAKTER VERISININ 1/
8'INI OKU
120 POKE12288+(8*CHAR)+BYTE,NUMBER:REM V
ERILERI BELLEGE SAKLA
140 NEXTBYTE,CHAR
150 PRINT"♥"TAB(255)CHR$(60)CHR$(61)TAB(
55)CHR$(62) CHR$(63): REM SHIFT+CLR/HOME
151 REM SATIR 150 YENI TANIMLANMIS KARAK
TERLERI EKRANA BASAR
170 GETA$:REM BIR TUSA BASIN
180 IFA$="" THEN 170:TUSA BASILINCAYA KA
DAR DONGU TEKRARLANIR
190 POKE53272,21:POKE53270,PEEK(53270)AN
D239:REM NORMAL KARAKTERLERE GECIS
200 DATA 129,37,21,29,93,85,85,85:REM 60
'INCI KARAKTER ICIN VERILER
210 DATA 66,72,84,116,117,85,85,85:REM 6
1'INCI KARAKTER ICIN VERILER
220 DATA 87,87,85,21,8,8,40,0:REM 62'INC
I KARAKTER ICIN VERILER
230 DATA 213,213,85,84,32,32,40,0:REM 63
'UNCU KARAKTER ICIN VERILER
240 END
```

GELİŞTİRİLMİŞ ZEMİN-RENGİ MODU

Geliştirilmiş zemin-rengi modu, her karakterin zemin ve ön plan renklerini kontrol etmenizi sağlayacaktır. Örneğin bu modda çalışırken beyaz bir ekranda sarı bir zemin üzerinde, mavi bir karakter görüntüleyebilirsiniz.

Geliştirilmiş zemin-rengi modu için 4 kayıt söz konusudur. Her kaydı 16 renkten herhangi birini elde edecek şekilde ayarlayabilirsiniz. Renk belleği, geliştirilmiş zemin-rengi modundaki ön plan renklerini tutmak için kullanılır. Renk kayıtlarının kullanımı tıpkı standart karakter modunda olduğu gibidir.

Geliştirilmiş karakter modu, kullanmak istediğiniz karakter çeşidinin sayısını sınırlar. Geliştirilmiş renk modu ile çalışırken, karakter ROM'undaki ilk 64 karakteri kullanabilirsiniz sadece. Bunun nedeni, karakter kodunun iki bitinin zemin rengini seçmekte kullanılmasıdır. Bu şöyle açıklanabilir:

"A" harfinin karakter kodu (POKE komutu ile ekrana yerleştireceğiniz sayı) 1'dir. Eğer geliştirilmiş renk modundaysanız, ekrana 1'i yerleştirdiğinizde, "A" harfini görürsünüz. Eğer 65 POKE ederseniz ve normal modda iseniz, karakter kodu (CHR$) 129 olan negatif "A" harfini görürsünüz.

Geliştirilmiş karakter modunda ise olay böyle işlemez. Bunun yerine ekranda önceki gibi negatif olmayan bir A harfi göreceksiniz, ancak zemin rengi değişik olacak. Aşağıdaki tablodan kodları bulabilirsiniz:

KARAKTER KODU			ARKA ZEMİN-RENK KAYDI	
SINIR	BİT 7	BİT 6	NUMARA	ADRES
0-63	0	0	0	53281-($D021)
64-127	0	1	1	53282-($D022)
128-191	1	0	2	53283-($D023)
192-255	1	1	3	53284-($D024)

Geliştirilmiş karakter madunu çalıştırabilmeniz, 53265 ($D011) adresinde bulunan VIC-II kaydının 6'ıncı bitini 1 yapmanızla mümkündür. Bu iş için, aşağıdaki POKE komutunu kullanabilirsiniz.

```
POKE 53265, PEEK(53265) OR 64
```

Geliştirilmiş renk modundan çıkabilmeniz ise yine aynı biti 0'a eşitlemenizle mümkündür. Yani şu yönergeyi kullanacaksınız:

```
POKE 53265, PEEK(53265) AND 191
```

BİT HARİTALAMA GRAFİKLERİ

Ya oyun yazarken ya da iş çizelgesi yaparken ya da herhangi bir programı yazarken, yani er geç günün birinde yüksek çözünürlüklü görüntüleri işlemeniz gerektiği sorunuyla karşılaşacaksınız.

Commodore 64'ün tasarımı bunu başaracak şekilde yapılmıştır: Yüksek-çözünürlük, bit haritalama yöntemi kullanılarak yapılabilir. Bit haritalamada, ekrandaki çözünürlüğün olası her noktası (piksel), bellekteki kendi bit (hane) değerini alır. Eğer noktanın bellek biti 1 ise nokta kullanılıyor demektir, yok eğer 0 ise, nokta kullanılmıyor demektir.

Yüksek çözünürlüklü grafik tasarımının 2 dezavantajı vardır, bu yüzden her zaman kullanılmaz. Birincisi: tüm ekranın bit haritasını çıkarmak bellekte çok fazla yer kaplar. Bunun nedeni; her noktanın bellekte kendisini kontrol eden bir bite sahip olması zorunluluğudur. Yani her piksel için bellekte 1 bitiniz (ya da 8 nokta için 1 baytınız) olması şarttır. Her karakterin 8'e 8'1ik olduğu, her satırda 25 karakterli 40 satırın olduğunu düşünürsek tüm ekranın çözünürlüğü 320*200 noktadır. Bu her birisinin bellekte bir bit gerektirdiği, 64000 ayrı nokta demektir. Yani tüm ekranın haritası 8000 baytlık bir bellek demektir.

Yüksek çözünürlüklü işlemler; genellikle kısa, basit ve tekrarlanan rutinlerden oluşur. Bu tip yüksek çözünürlüklü rutinleri BASIC'le yazmaya uğraşırsanız maalesef işin oldukça yavaş gittiğini göreceksiniz. Çözüm için ya tamamıyla makina dilini kullanmanız ya da BASIC programınızdan SYS komutunu kullanarak bu yüksek çözünürlüklü rutinleri çağırmanız gerekir.

Bu bölümdeki örnekler, daha açıklayıcı olmaları için, BASIC ile yazılmıştır. Şimdi teknik ayrıntılarına geçelim.

BİT HARİTALAMA

Bilgisayar dünyasının en popüler grafik yapma tekniklerinden birisidir. Bit haritalama yöntemi ile oldukça detaylı resimler elde edebilirsiniz. Commodore 64 bit haritalama modunda ise, ekranda belleğin 8K'lık bir bölümünü direkt olarak görebilirsiniz. Bu, ekrandaki her noktanın kullanılıp kullanılmadığını denetleme yapabilmeniz demektir.

Commodore 64'te 2 tür bit haritalama modu vardır:

1. Standart (yüksek çözünürlüklü) mod (320 * 200 nokta çözünürlük)
2. Çok-renkli mod (160 * 200 nokta çözünürlük)

Bunlar, karakter tiplerine çok benzerler. Örneğin standart modun çözünürlüğü daha yüksektir, ama renk seçimi azdır. Buna karşılık çok-renkli modda 8*8 noktalık karede çok sayıda renk seçme olanağına sahipsiniz.

STANDART YÜKSEK ÇÖZÜNÜRLÜKLÜ BİT HARİTALAMA MODU

Standart bit haritalama modu size 320 yatay, 200 dikey nokta çözünürlüğü sağlayacaktır. Her 8*8'lik bölüm için ise 2 renk seçme şansınız var. Bit haritalama moduna geçmek istiyorsanız VIC-II kontrol kayıtlarının 53265 ($D011) adresinde bulunan 5'inci bite 1 yerleştirmelisiniz. Bunu aşağıdaki POKE komutu ile yapabilirsiniz:

```
POKE 53265, PEEK(53265) OR 32
```

Bu moddan çıkmak için de aynı bite 0 yerleştirmeniz gerekir:

```
POKE 53265, PEEK(53265) AND 223
```

Bu haritalama modunun ayrıntılarına geçmeden önce değinmemiz gereken bir konu daha var: Bit haritasını nereye yerleştirelim?

NASIL ÇALIŞIR?

Hatırlayacağınız gibi PROGRAMLANABİLİR KARAKTERLER bölümünde RAM'da depolanan bir karakterin bit kümesini istediğiniz gibi oluşturabileceğinizi anlatmıştık. Eğer ekranda görüntülenen karakteri değiştirmek isterseniz, tek bir noktayı değiştirerek neler olacağını da izleyebilirsiniz. Bu, bit haritalamanın temelidir. Tüm ekran programlanabilir karakterlerle doludur ve siz değişikliklerinizi doğrudan, karakterlerin biçimlerini aldıkları bellekte yapabilirsiniz.

Hangi karakterin görüntüleneceğini kontrol eden ekran belleği yerleşimleri, şimdi artık renk bilgileri için kullanılmaktadır. Örneğin; 1024 yerleşimine 1 POKE 'lamanız artık sol üst köşede bir "A" harfi görüntüsünü değil orada bulunan birlerin rengini kontrol edecektir.

Karakter modunda renkleri renk belleğinden alınıyordu. Oysa bit haritalama modunda renkler ekran belleğinden alınır.

Grafik ekranındaki 8*8'lik alanda yer alan bir bitin değeri eğer 1 ise bu bitin rengini, ekran belleğinin 4 üst biti verir. O değerinde olan bitler ise renklerini 4 alt bitten alırlar.

ÖRNEK: Şu satırları yazın:

```
5 BASE=2*4096: POKE53272, PEEK(53272) OR
 8: REM 8192'E BIT HARITASINI KOY
10 POKE53265, PEEK(53265) OR 32: REM BIT
 HARITA MODUNU GIR
```

Şimdi programı işletin (RUN).

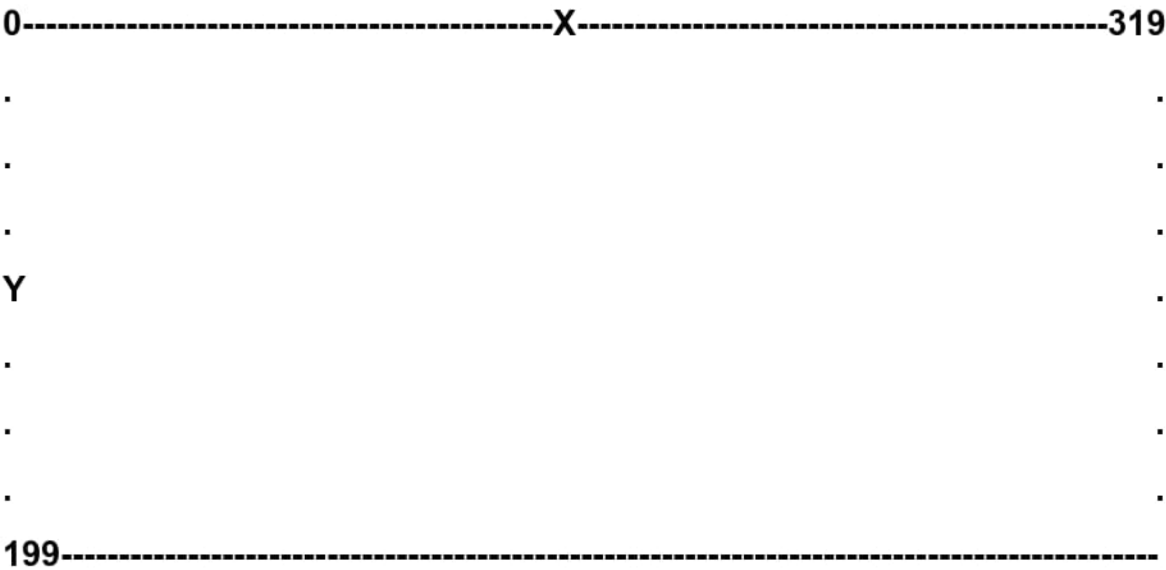
Ekrana bir sürü saçma noktalar çıktı, değil mi? Tıpkı normal ekran modunda olduğu gibi. YÜKSEK ÇÖZÜNÜRLÜKLÜ (HI-RES) ekranın da kullanılmadan önce temizlenmesi gerekir. Bu durumda, maalesef **CLR** tuşu çalışmaz. Bunun yerine yapacağınız; programlanabilir karakterleriniz için kullandığınız belleği temizlemektir. Bunun için önce **RUN/STOP** ve **RESTORE** tuşlarına basın, daha sonra ise aşağıdaki satırları programınıza ekleyin. Ekranın temizlendiğini göreceksiniz.

```
20 FOR I = BASE TO BASE + 7999: POKE I,
0: NEXT: REM BIT HARITASINI TEMIZLE
30 FOR I = 1024 TO 2023: POKE I, 3: NEXT
: REM RENGI TURKUVAZ VE SIYAH YAP
```

Şimdi programınızı tekrar işletin (RUN yazın). Tüm ekranın temizlendiğini ve yeşilimsi bir mavi ile kaplandığını göreceksiniz.

Şimdi yapacağınız ise ekran üzerindeki noktaların YÜKSEK ÇÖZÜNÜRLÜKLÜ ekran üzerinde görünmesini ve kaybolmasını sağlamaktır. Bir noktayı kullanmak için: öncelikle bilmeniz gereken, karakter belleğinde 1 değerini vereceğiniz bitin doğru yerini nasıl bulacağınızdır. Başka bir deyişle: Önce değiştirmek istediğiniz karakteri; yani karakterin bulunduğu satırı ve hangi bit olduğunu bulmak zorundasınız.

Bunu bulmak için bir formüle ihtiyacınız var. Noktanın yatay ve dikey konumları için X ve Y kullanacağız. X = 0 ve Y = 0 konumunda bulunan nokta, ekranın sol üst köşesindeki noktadır. Sağa doğru ilerledikçe noktaların X değeri, aşağı doğru inildikçe de noktaların Y değeri artar. Bit haritalama yöntemini kullanmanın en iyi yolu, şöyle bir görüntü düzeni kurmaktır:



Her noktanın bir X ve Y koordinatı olacağı için, bu formatta ekrandaki bütün noktaları kontrol etmek oldukça kolay.

Ancak aslında şöyle bir formatla karşı karşıyasınız:

.....	BYTE 0	BYTE 8	BYTE 16	BYTE 24	.....	BYTE 312
	BYTE 1	BYTE 9	.	.		BYTE 313
	BYTE 2	BYTE 10	.	.		BYTE 314
	BYTE 3	BYTE 11	.	.		BYTE 315
	BYTE 4	BYTE 12	.	.		BYTE 316
	BYTE 5	BYTE 13	.	.		BYTE 317
	BYTE 6	BYTE 14	.	.		BYTE 318
.....	BYTE 7	BYTE 15	.	.		BYTE 319
.....	BYTE 320	BYTE 328	BYTE 336	BYTE 344	.....	BYTE 632
	BYTE 321	BYTE 329	.	.		BYTE 633
	BYTE 322	BYTE 330	.	.		BYTE 634
	BYTE 323	BYTE 331	.	.		BYTE 635
	BYTE 324	BYTE 332	.	.		BYTE 636
	BYTE 325	BYTE 333	.	.		BYTE 637
	BYTE 326	BYTE 334	.	.		BYTE 638
.....	BYTE 327	BYTE 335	.	.		BYTE 639

Bit haritalarını oluşturan programlanabilir karakterlerin her biri 25 satır ve 40 sütun içinde düzenlenebilir. Bu, bir metnin düzenlenmesini kolaylaştırmakla birlikte, bit haritalama yöntemini biraz güçleştirir. (KARIŞIK MODLAR bölümünde bu daha geniş ve ayrıntılı açıklanacaktır).

Aşağıda vereceğimiz formül, bit haritası ekranındaki bir noktanın daha kolay kontrol edilmesini sağlamak için, yeterli olacaktır.

Görüntü belleği alanının başlangıcı TABAN olarak adlandırılır. Noktanızın (0'dan 24'e kadar) satır no.'su:

SATIR = INT(Y/8) (Her satırda 320 bayt vardır.)

Bu satırdaki (0'dan 39'a) karakter konumu hesaplanması ise şöyle yapılabilir:

KAR = INT(X/8) (Her karakter için 8 bayt vardır.)

Karakter konumunun (0'dan 7'ye) sırası ise:

SIRA = Y AND 7

Bu baytın biti de şu formülle hesaplanır:

BİT = 7-(X AND 7)

Şimdi bu formülleri birleştirelim. Artık karakter belleği noktasının (X, Y) yer aldığı baytı elde edebiliriz.

BYTE= TABAN + SATIR * 320 + KAR * 8 + SIRA

X, Y tablomuzda X ve Y koordinatlarındaki birbiri görünür duruma getirmek için şu satırı yazın:

```
POKE BYTE, PEEK(BYTE) OR 2 ↑ BIT
```

Şimdi bu hesaplamaları programımıza ekleyelim. Commodore 64 sizin için bir sinüs eğrisi çizmeye hazırdır. Şu programı yazın:

```
5 BASE=2*4096:POKE53272,PEEK(53272)OR8
10 POKE 53265,PEEK(53265)OR32
20 FORI=BASETOBASE+7999:POKEI,0:NEXT
30 FORI=1024TO2023:POKEI,3:NEXT
50 FORX=0TO319STEP.5
60 Y=INT(90+80*SIN(X/10))
70 CH=INT(X/8)
80 RO=INT(Y/8)
85 LN=YAND7
90 BY=BASE+RO*320+8*CH+LN
100 BI=7-(XAND7)
110 POKEBY,PEEK(BY)OR(2↑BI)
120 NEXTX
125 POKE1024,16
130 GOTO130
```

Satır 60'ta yer alan hesaplama, sinüs fonksiyonunun değerini (+1, -1) aralığından (10 ile 170) aralığına taşıyacaktır. 70 ve 100 arasındaki satırlar, karakter, sıra, bayt ve istenen biti hesaplamaktır. Aradaki bölüm karakter, sıra, bayt ve istenen biti hesaplamaktır. Bu hesaplamalarda yukarıda verdiğimiz formül kullanılır. Satır 125 ekranın sol üst köşesinin rengini değiştirmekle programı bitirir. Satır 130 ise, programı sonsuz bir döngüye sokar. Görüntüye bakmaktan bıkarsanız, eliniz **RUN/STOP** tuşunun üzerindeyken **RESTORE** tuşuna basın.

Sinüs eğrisinin programına ekleyeceğiniz birkaç satırla bir yarım daire görüntüsü elde edebilirsiniz. Bunun için şu satırları yazın:

```
50 FORX=0TO160
55 Y1=100+SQR(160*X-X*X)
56 Y2=100-SQR(160*X-X*X)
60 FORY=Y1TOY2STEPY1-Y2
70 CH=INT(X/8)
80 RO=INT(Y/8)
85 LN=YAND7
90 BY=BASE+RO*320+8*CH+LN
100 BI=7-(XAND7)
110 POKEBY,PEEK(BY)OR(2↑BI)
114 NEXT
```

Böylece, ekranın yüksek çözünürlüklü alanında bir yarım-daire elde etmiş olursunuz.

NOT: BASIC değişkenleri, sizin yüksek çözünürlüklü ekranınızın üzerinde yer alabilirler. Daha geniş bir bellek alanı kullanmak istiyorsanız: BASIC'in alt kısmını yüksek çözünürlüklü ekranın üst kısmına taşıyın. Makina dilinde bu tip sorunlarınız olmaz. Bu, sadece programlarınızı BASIC dilinde yazıyorsanız sorun çıkartır.

ÇOK-RENKLİ BİT HARİTALAMA MODU

Bu mod, çok renkli karakter moduna benzer. Bit haritasının her 8*8'lik bölümünü, 4 değişik renkte görüntülemeniz mümkündür. Ve yine çoklu- karakter modunda olduğu gibi yatay çözünürlük, 320 noktadan 160 noktaya iner.

Çok-renkli bit harita modunda belleğin 8K'lık bölümünü kullanırsınız. Bu, bit haritası içindir. Renklerinizi ise:

1- 0 numaralı zemin rengi kayıtlarından (ekranın zemin rengi).
2- Video-matrisinden (4 üst bit, bir olası rengi, 4 alt bit diğerini verir).
3- Renk belleğinden seçebilirsiniz.

Çok-renkli bit harita moduna geçebilmek için 53265 ($D011) adresindeki 5'inci biti ve 53270 ($D016) adresindeki 4'üncü biti 1 yapmanız gerekir. Bunu şu POKE komutu ile sağlayabilirsiniz.

```
POKE53265,PEEK(53265)OR 32:POKE53270,PEE
K(53270)OR16
```

Çok-renkli bit harita modundan çıkmak için ise yine aynı bitleri 0 yapmanız gerekir. Şu POKE yönergesini kullanabilirsiniz:

```
POKE53265,PEEK(53265)OR 32:POKE53270,PEE
K(53270)AND239
```

Tıpkı standart (yüksek çözünürlüklü) bit harita modunda olduğu gibi burada da görüntüleme için kullanılan belleğin 8K'lık bölümü ile ekranda görünen arasında bire bir eşleme söz konusudur. Ancak, yatay noktalar iki bit genişliğindedir. Görüntüleme belleğindeki her 2 bit bir nokta oluştururlar. Bu bitlerle elde edebileceğiniz renk sayısı 4 'tür.

BİT ÇİFTİ	RENK BİLGİSİNİN GELİŞ YERİ
00	Zemin rengi no. 0 (ekran rengi)
01	Ekran belleğinin 4 üst biti
10	Ekran belleğinin 4 alt biti
11	Renk yarı baytı (yarı-bayt=1/2 bayt=4 bit)

DÜZ KAYDIRMA (SMOOTH SCROLLING)

VIC-II çipi, düz kaydırmayı hem yatay hem de dikey yönlerde destekler. Düz kayma tüm ekranın tek yöndeki noktasal hareketidir. Hareketin yönü sola, sağa, yukarı ya da aşağı olabilir. Ekrandaki karakterler düzgün bir şekilde diğer tarafa kayarken ekranda yeni karakterler görünmeye başlar.

VIC-II çipi birçok işi sizin yerinize yapmakla birlikte, asıl kaydırma işlemi, makina dili bir programla yapılmalıdır. VIC-II çipi, video ekranını 8 yatay ve 8 dikey konumdan herhangi birisine yerleştirebilme yetisindedir. Yerleştirme, VIC-II kaydırma kayıtları tarafından kontrol edilir. Bunun yanı sıra VIC-II çipi, 38 sütun, 24 sıra moduna sahiptir.

Düz kaydırma işlemini şöyle sıralayabiliriz:

1- Ekranı daraltın (kenarlar genişleyecektir).
2- Kaydırma kayıtlarını maksimum (veya kayma yönüne bağlı olarak minimum) değerine getirin.
3- Yeni verileri ekranın uygun olan kısmına yerleştirin.
4- Kaydırma kayıtlarını maksimum (veya minimum) değere kadar artırın (veya azaltın).
5- Bu noktada makina dili rutinini kullanarak, tüm ekranı bir karakter kadar kayma yönünde hareket ettirin.
6- 2'nci şıkka geri dönün.

38 sütun moduna geçebilmeniz için 53270 ($D016) adresindeki 3'üncü bitin 0 değerini almış olması gerekir. Bu iş için aşağıdaki POKE yönergesini kullanın:

```
POKE 53270, PEEK(53270) AND 247
```

40 sütun moduna geri dönebilmeniz için ise aynı biti 1 yapmanız gerekli. Bu durumda, şu POKE yönergesini kullanacaksınız:

```
POKE 53270, PEEK(53270) OR 8
```

24 sıra moduna mı girmek istiyorsunuz? 53265 ($D011) adresindeki 3'üncü biti 0 yapın. Yani şu POKE yönergesini kullanın:

```
POKE 53265, PEEK(53265) AND 247
```

Şimdi 25 sıra moduna geri dönmek istiyorsunuz. Yine aynı biti 1 yapın. Aşağıdaki POKE yönergesi bunu yapacaktır:

```
POKE 53265, PEEK(53265) OR 8
```

X yönünde kaymayı sağlamak için VIC-II çipini 38 sütun moduna geçirmeniz gerekir. Bu, yeni verilere, kayacakları alanı sağlar. SOLA kaydıracaksanız, yeni veri sağda yer almalıdır. Aynı şekilde SAĞA kaydıracaksanız, yeni verinizin bu kez de solda bulunması gerekir. Lütfen ekran belleğinizde hala 40 sütun olmasına rağmen, sadece 38 sütunun görünebildiğine dikkat edin.

Y yönünde kaymasını istiyorsanız şimdi de VIC-II çipini 24 sıra moduna geçireceksiniz. Yukarı doğru kaydırırken, yeni yeriniz SON sırada, aşağı doğru kaydırmada da yeni yeriniz İLK sırada bulunmaktadır. X yönündeki kaydırmada ekranın her yanı doluyken, şimdi sadece bir alanın dolu olduğunu göreceksiniz. Y kaydırma kaydına 0 değerini verin, böylece ilk sırayı doldurmuş olacaksınız, artık bilgisayarınız yeni veri için hazırdır. Y kaydırma kaydına 7 değerini verdiğiniz anda ise son sıra dolacaktır.

X yönündeki kaydırmalar için kullanılan kayıt, 53270 ($D016) adresindeki VIC-II kontrol kaydının 2 ve 0'ıncı bitleri arasında yer alır. Bu noktada tek önemli olan, sadece bu bitlerin etkilenmesidir. Aşağıdaki POKE komutu bunu sağlar:

```
POKE 53270, (PEEK(53270) AND 248)+X
```

X: Ekranın 0'dan 7'ye kadar olan X konumudur.

Y yönündeki kaydırmalar için ise 53265 ($D011) adresindeki VIC-II kontrol kaydının 2 ve 0'ıncı bitleri arasında yer alan kayıt kullanılır. Daima bu bitler üzerinde çalışmalısınız. Bunun için de şu POKE yönergesini kullanın:

```
POKE 53265, (PEEK(53265) AND 248) + Y
```

Y: Ekranın 0'dan 7'ye kadar olan Y konumudur.

Ekrandaki metni aşağıdan yukarıya kaydırmak için, 53265 adresindeki 0 ile 7 arasındaki bitlerin en alt 3 bitini kullanın, yeni verileri ekranın altındaki kapalı alana girin ve işlemi tekrar edin.

Ekran üzerindeki karakteri soldan sağa kaydırmak için ise 53270 adresindeki 0 ve 7 arasındaki bitlerin en alt 3 bitini kullanın, yeni verileri 0'ıncı kolona yazın ya da POKE komutu ile yerleştirin ve işlemi tekrar edin.

Kaydırma bitleri için -1 kullandıysanız, ekrandaki metin zıt yönde hareket edecektir. Aşağıdaki örnek ekranda aşağı doğru kayma yapar.

```
10 POKE 53265, PEEK(53265) AND 247
20 PRINT CHR$(147)
30 FOR X=1 TO 24: PRINT CHR$(17);: NEXT
40 POKE 53265, (PEEK(53265) AND 248) + 7
: PRINT
50 PRINT"     HELLO";
60 FOR Y= 6 TO 0 STEP -1
70 POKE 53265, (PEEK(53265) AND 248) +Y
80 FOR X= 1 TO 50: NEXT
90 GOTO 40
```

YARATIKLAR (SPRITES)

"Yaratıklar", kullanıcı tarafından tanımlanabilen özel tipte karakterlerdir ve ekranın her yerinde görüntülenebilirler. Yaratıkların tüm kontrolünü VIC-II çipi sağlar. Sizin yapacağınız tek şey, yaratığın neye benzeyeceğini, renginin ne olacağını ve nerede görüneceğini belirtmektir. Bırakın gerisini VIC-11 çipi yapsın! Yaratıklarınız için 16 renk kullanabilirsiniz. Yaratıkları istediğiniz grafik modunda kullanabilirsiniz: bit haritalama, karakter, çok-renkli, vs... Hangi modda kullanırsanız kullanın, yaratıkların şekli değişmeyecektir. Her yaratık kendi renk tanımını, kendi modunu (yüksek çözünürlüklü veya çok-renkli) ve kendi şeklini içerir.

VIC-II çipi bir kere de 8 yaratığı birden kullanmanızı otomatik olarak sağlar. Eğer çok sayıda yaratık istiyorsanız RASTER KESİNTİ tekniklerini kullanabilirsiniz.

YARATIKLARIN özelliklerini şöyle sıralayabiliriz:

1-24 yatay, 21 dikey nokta boyutu.
2-Her yaratık için ayrı renk kontrolü.
3-Çok-renkli yaratık modu.
4-Yatay, dikey ya da her iki yönde de 2 misli büyütme.
5-Yaratığa zemine göre öncelik verme.
6-Yaratığın diğer yaratığa göre sabit önceliği.
7-Yaratık çarpışmalarının kontrolü.
8-Yaratıkların zeminle çarpışmalarının kontrolü.

Yaratıkların bu özel yetenekleri heyecanlı oyunlar programlamanızı oldukça kolaylaştıracaktır. Yaratıkların kaynağının bilgisayarınızın donanımı (hardware) olmasına rağmen, BASIC'te de kaliteli oyun yazmak mümkündür.

VIC-II çipinin dolaysız olarak desteklediği 8 tane yaratık vardır. Bunlar 0'dan 7'ye kadar numaralanmışlardır ve her biri kendi tanımına, konum kaydına ve renk kaydına sahiptir. Bunun yanı sıra her birinin görüntülenmesi ve çarpışmaları kendi bitleri ile kontrol edilir.

YARATIK TANIMLAMA

Yaratıkları da tıpkı programlanabilir karakterler gibi tanımlarız. Ancak yaratıklar için gereksindiğiniz bayt sayısı daha fazladır, çünkü bunların boyutları programlanabilir karakterlerden daha büyüktür. Bir yaratık 24*21, yani 504 noktadan oluşur. Bu 63 bayt (504/8) demektir. 63 bayt, her bir sırada 3 bayt olmak üzere 21 sıra olarak düzenlenmiştir.

KOLON NUMARASI	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23
BİT	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
BİT VERİ DEĞERLERİ AÇIK=1*DEĞ.	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1
SATIR 0																								
SATIR 1																								
SATIR 2																								
SATIR 3																								
SATIR 4																								
SATIR 5																								
SATIR 6																								
SATIR 7																								
SATIR 8																								
SATIR 9																								
SATIR 10																								
SATIR 11																								
SATIR 12																								
SATIR 13																								
SATIR 14																								
SATIR 15																								
SATIR 16																								
SATIR 17																								
SATIR 18																								
SATIR 19																								
SATIR 20																								

ŞEKİL 3-2. YARATIK TANIMLAMA BLOĞU.

Bir yaratığın tanımını şöyle gösterebiliriz:

BAYT 0	BAYT 1	BAYT 2
BAYT 3	BAYT 4	BAYT 5
BAYT 6	BAYT 7	BAYT 8
..	..	..
..	..	..
..	..	..
BAYT 60	BAYT 61	BAYT 62

Yaratıkların oluşmasını, bit olarak tanımlama blokları şeklinde de inceleyebiliriz. Şekil 3-2, bu konuyu anlamanıza yardımcı olacaktır.

Standart (yüksek çözünürlük) yaratıklar da 1 olan her bit, yaratığın ön plan renginde görüntülenmesini sağlar. 0 değerindeki bitler ise saydamdır ve arkalarında ne varsa görüntülerler. Bu özellikler ile standart karakterlere benzerler.

Çok-renkli yaratıklar, çok-renkli karakterlere benzerler. Ekstra renk çözünürlüğü için yatay çözünürlük kullanılır. Böylece bir yaratığın çözünürlüğü 12 yatay nokta, 21 dikey nokta olur. Bir yaratıktaki her noktanın genişliği 2 kat artarken, görüntülenebilir renk sayısı da 4 olur.

YARATIK GÖSTERGEÇLERİ

Bir yaratığın tanımlanabilmesi için 63 bayt gerektiğini söylemiştik. Ancak her yaratığın sonunda, yer tutucu işlevini yerine getirecek bir bayt daha gereklidir. Böylece her yaratık 64 baytlık yer kaplamış olur. 64 'ün çift bir sayı ve ikili aritmetikte çift bir kuvvet olması yaratığın bellekteki yerini hesaplamanız, kolaylaştıracaktır.

8 yaratığın her birinin **YARATIK GÖSTERGEÇİ** olarak kullanılan bir baytı vardır. Yaratık göstergeçleri yaratık tanımlarının bellekte nereye yerleştirildiklerini kontrol ederler. Bu 8 bayt her zaman ekran belleğinin 1K'lık kısmında son 8 bayt olarak yerleştirilir. Bu kısım COMMODORE 64'te normalde, 2040 ($07F8) adresinden başlar. Ancak ekranı hareket ettirmeniz, yaratık göstergeçlerinin de hareket etmesine neden olacaktır.

Yaratık göstergeçlerinin her biri 0'la 255 arasındaki bir sayıyı içerebilir. Bu sayı, yaratığın tanımıdır. Her yaratık 64 baytla tanımlandığına göre, bir göstergeç VIC-II çipinin erişebileceği bellekte yer alan 16K'lık bloğun herhangi bir yerini "görebilir" (çünkü:256 * 64 = 16K).

Şimdi 2040 adresinde yer alan 0 numaralı yaratık göstergecinin 14 sayısını taşıdığını düşünelim. Bunun anlamı, o yaratık göstergecinin kaset tamponundaki 14*64=896 adresinden başlayan 64 bayt ile görüntüleneceğidir. Aşağıdaki formül, sorunu daha anlaşılır kılacaktır:

KONUM = (KÜME * 16384) + (YARATIK GÖSTERGECİ DEĞERİ * 64)

Küme, VIC-II çipinin gördüğü 16K'lık bölümdür ve 0'la 3 arası bir değere sahiptir.

Bu formül yaratık tanımının yer aldığı 64 baytlık bloğun başlangıcını verir.

Daha önce belirtmiştik; VIC-II çipinin KÜME 0'a veya KÜME 2'ye bakması halinde, belirli adreslerde bulunan karakter seti bir ROM görüntüsüne sahip demektir. Yaratık tanımlarını buraya yerleştiremeyiz. Eğer 128'den fazla yaratık tanımlamak istiyorsanız diğer kümelerden birini, yani, 1 ya da 3'ü seçin.

YARATIK ETKİNLEŞTİRME

53269 ($D015) adresindeki VIC-II kontrol kaydı YARATIK ETKİN (SPRITE ENABLE) kaydı olarak bilinir. Yaratıkların görünmesini ya da kaybolmasını sağlayan bu kayıt, her yaratık için bir bit içerir. Kayıt görünümü şöyledir:

$D015   7 6 5 4 3 2 1 0

Örneğin birinci yaratığın görünmesi için, 0 bitin 1 değerini almasını sağlayın. Şu POKE yönergesini kullanın:

```
POKE 53269, PEEK(53269) OR 2
```

Yönergeyi biraz daha genişleterek şöyle yazabilirsiniz:

```
POKE 53269, PEEK(53269) OR (2↑SN)
```

(SN: 0'dan 7'ye kadar olan yaratık numarası.)

NOT: Bir yaratığın görünmeden önce işler duruma getirilmesi gerekir.

YARATIK DEVRE DIŞI BIRAKMA

Bir yaratığın kaybolmasını istiyorsanız, o yaratığın 53269 ($D015) adresindeki VIC-II kontrol kaydının uygun bitini 0 yapmak zorundasınız. Aşağıdaki POKE yönergesi bunu sağlayacaktır.

```
POKE 53269, PEEK(53269) AND (255-2↑SN)
```

(SN: 0'dan 7'ye kadar olan yaratık numarası.)

RENKLER

Bir yaratık için 16 renk olasılığı vardır. Bu renkler VIC-II çipi tarafından üretilirler. Her yaratığın kendine ait bir yaratık renk kaydı bulunur. Aşağıda gördüğünüz, renk kayıtlarının bellekteki adresleridir:

ADRES		AÇIKLAMA
53287	($D027)	YARATIK 0 RENK KAYDI
53288	($D028)	YARATIK 1 RENK KAYDI
53289	($D029)	YARATIK 2 RENK KAYDI
53290	($D02A)	YARATIK 3 RENK KAYDI
53291	($D02B)	YARATIK 4 RENK KAYDI
53292	($D02C)	YARATIK 5 RENK KAYDI
53293	($D02D)	YARATIK 6 RENK KAYDI
53294	($D02E)	YARATIK 7 RENK KAYDI

Bir yaratıktaki tüm noktalar, yaratık renk kaydındaki renge göre görüntülenir. Yaratığın geri kalan kısımları ise saydam olacak ve yaratığın arkasında ne varsa onu gösterecektir.

ÇOK-RENKLİ MOD

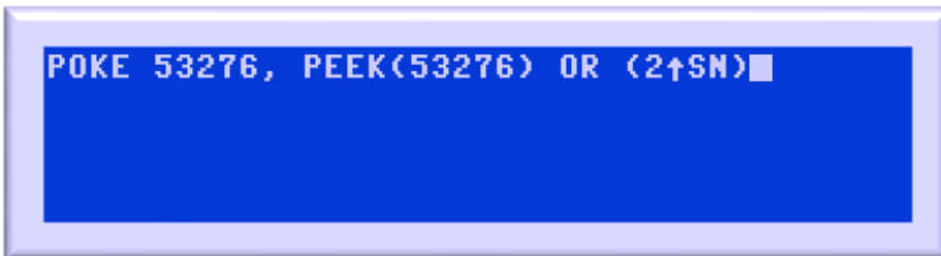
Çok-renkli modla bir yaratık için, 4 değişik renk kullanma olanağını elde edebilirsiniz. Ancak, tıpkı diğer çok-renkli modlarda olduğu gibi, burada da yatay çözünürlük olanağı, yarı yarıyadır. Yani: çok-renkli yaratık modunda çalışıyorsanız, 24 nokta değil, 12 çift nokta kullanabilirsiniz. Her nokta çifti, BİT ÇİFTİ olarak adlandırılır. Yaratıklarınıza renk seçme aşamasındaysanız, her nokta çiftini tek bir nokta gibi düşünmelisiniz. Yaratığınız için seçtiğiniz renkleri verebilmeniz için, nasıl bir bit çifti kullanmanız gerektiğini, aşağıda verdiğimiz tablodan bulabilirsiniz.

BİT ÇİFTİ	AÇIKLAMA
00	SAYDAM, EKRAN RENGİ
01	0 NUMARALI ÇOK-RENKLİ YARATIK KAYDI (53285) ($D025)
10	YARATIK RENK KAYDI
11	1 NUMARALI ÇOK-RENKLİ YARATIK KAYDI (53286) ($D026)

NOT: Yaratık ön plan rengi 10, karakter ön plan ise 11'dir.

YARATIKLARI ÇOK-RENKLİ MODDA OLUŞTURMA

Çok-renkli modda bir yaratık istiyorsanız 53276 ($D01C) adresinde bulunan VIC-II kontrol kaydını çalıştırmalısınız. Aşağıdaki POKE yönergesi bunu sağlayacaktır.



```
POKE 53276, PEEK(53276) OR (2↑SN)
```

(SN: 0'dan 7'ye kadar olan yaratık numarası.)

Eğer yaratığınız çok-renkli moddan kurtulsun istiyorsanız yine aynı VIC-II kontrol kaydını kapatın. Aşağıdaki POKE yönergesini kullanabilirsiniz:

```
POKE 53276, PEEK(53276) AND (255-↑SN)
```

(SN: 0'dan 7'ye kadar olan yaratık numarası.)

YARATIK GENİŞLETME

VIC-II çipinin bir başka yeteneği de: bir yaratığı dikey, yatay ya da hem dikey hem de yatay olarak genişletebilmesidir. Yaratıktaki bir noktayı genişletmekle, 2 kat daha geniş ya da 2 kat daha uzun bir nokta elde edebilirsiniz. Aslında çözünürlük artmamakta, sadece yaratığınız büyümektedir.

Bir yaratığı yatay olarak genişletmek istiyorsanız. VIC-II kontrol kaydının 53277 ($D01D) adresindeki uygun bitin 1 değerini almasını sağlamalısınız. Aşağıdaki POKE yönergesini kullanırsanız yaratığınızı X yönünde genişletmiş olacaksınız:

```
POKE 53277, PEEK(53277) OR (2↑SN)
```

(SN: 0'dan 7'ye kadar olan yaratık numarası.)

Yatay yönde sağladığınız genişlemeyi VIC-II kontrol kaydının 53277 ($D01D) adresindeki biti 0 yaparak önleyebilirsiniz. Aşağıdaki POKE yönergesini kullanın.

```
POKE 53277, PEEK(53277) AND(255-2↑SN)
```

(SN: 0'dan 7'ye kadar olan yaratık numarası.)

Bir yaratığı dikey olarak genişletmenin yolu ise VIC-II kontrol kaydının 53271 ($D017) adresinde yer alan bite 1 değerini vermektir. Aşağıdaki POKE yönergesi yaratığın dikey yönde genişlemesini sağlayacaktır:

```
POKE 53271, PEEK(53271) OR (2↑SN)
```

(SN: 0'dan 7'ye kadar olan yaratık numarası.)

Eğer yaratığınızın yine eski halini almasını istiyorsanız aynı bite 0 değerini verin. Aşağıdaki POKE yönergesi yaratığınızı yine eski haline döndürecektir:

```
POKE 53271, PEEK(53271) AND (255-2↑SN)
```

(SN: 0'dan 7'ye kadar olan yaratık numarası.)

YARATIKLARI KONUMLANDIRMA

Bir yaratığı oluşturdunuz. Şimdi de ekranda hareket etmesini istiyorsunuz. Commodore 64'ünüz bunun için şu 3 kaydı kullanacaktır:

1. YARATIĞIN X KONUM KAYDI

2. YARATIĞIN Y KONUM KAYDI

3. YARATIĞIN X KONUM KAYDI EN SOL BİTİ (ESB)

Her hareketli grafiğin bir X konumu kaydı, bir Y konumu kaydı ve X en önemli bit kaydında bir biti vardır. Bu, karakterlerinizi çok doğru bir şekilde konumlandırmanıza olanak tanır. Yaratığınızı 512 olası X konumuna ve 256 olası Y konumuna yerleştirebilirsiniz.

X ve Y konum kayıtları, bir ekip gibi birlikte çalışırlar. X ve Y kayıtlarının yerleri bellek haritasında şu durumdadırlar: Önce 0 numaralı yaratık için X kaydı, sonra yine aynı yaratık için Y kaydı. İkinci olarak önce 1 numaralı yaratık için X konumu, sonra yine aynı yaratık için Y konumu. Bu böylece sürer gider. 16 tane X ve Y kayıtlarının hepsinden sonra, X konumunun (X ESB) en solda yer alan biti kendi kaydına yerleşir.

Aşağıdaki çizelge her bir yaratığın konum kaydını göstermektedir. POKE yönergesini kullanarak tüm konumları zamanında kullanabilirsiniz:

KONUM		AÇIKLAMA
ONLU	ONALTILI	
53248	($D000)	YARATIK 0 X KONUM KAYDI
53249	($D001)	YARATIK 0 Y KONUM KAYDI
53250	($D002)	YARATIK 1 X KONUM KAYDI
53251	($D003)	YARATIK 1 Y KONUM KAYDI
53252	($D004)	YARATIK 2 X KONUM KAYDI
53253	($D005)	YARATIK 2 Y KONUM KAYDI
53254	($D006)	YARATIK 3 X KONUM KAYDI
53255	($D007)	YARATIK 3 Y KONUM KAYDI
53256	($D008)	YARATIK 4 X KONUM KAYDI
53257	($D009)	YARATIK 4 Y KONUM KAYDI
53258	($D00A)	YARATIK 5 X KONUM KAYDI
53259	($D00B)	YARATIK 5 Y KONUM KAYDI
53260	($D00C)	YARATIK 6 X KONUM KAYDI
53261	($D00D)	YARATIK 6 Y KONUM KAYDI
53262	($D00E)	YARATIK 7 X KONUM KAYDI
53263	($D00F)	YARATIK 7 Y KONUM KAYDI
53264	($D010)	YARATIK X ESB KAYDI

Bir yaratığın konumu, yaratığın tanımlanacağı, 24*21 noktalık alanın SOL ÜST köşesinden hesaplanabilir. Yaratığınızı oluştururken kaç nokta kullandığınız hiç önemli değildir. Yaratık olarak sadece 1 noktayı kullansanız ve bunu ekranın tam ortasında isteseniz bile, konum hesabına sol üst köşeden başlamak zorundasınız.

DİKEY KONUMLANDIRMA

X yönündeki konumlamalar, dikey konumlamalardan biraz daha zordur. Biz önce Y yönündeki konumlamalardan söz edeceğiz.

Ekranda, Y yönünde, her birini tek tek programlayabileceğiniz 200 nokta vardır. Yaratıkların Y konumlama kayıtları 255'e kadar olan sayıları içerebilir. Bu; bir yaratığı aşağı ve yukarı hareket ettirirken, gerekenden daha fazla kayıt yerleşimine sahip olduğunuz anlamına gelir. Bu arada yaratığınızın ekrandan yavaşça çıkıp, kaybolmasını da isteyebilirsiniz. Bunun için de 200'den fazla değere sahip olmalısınız.

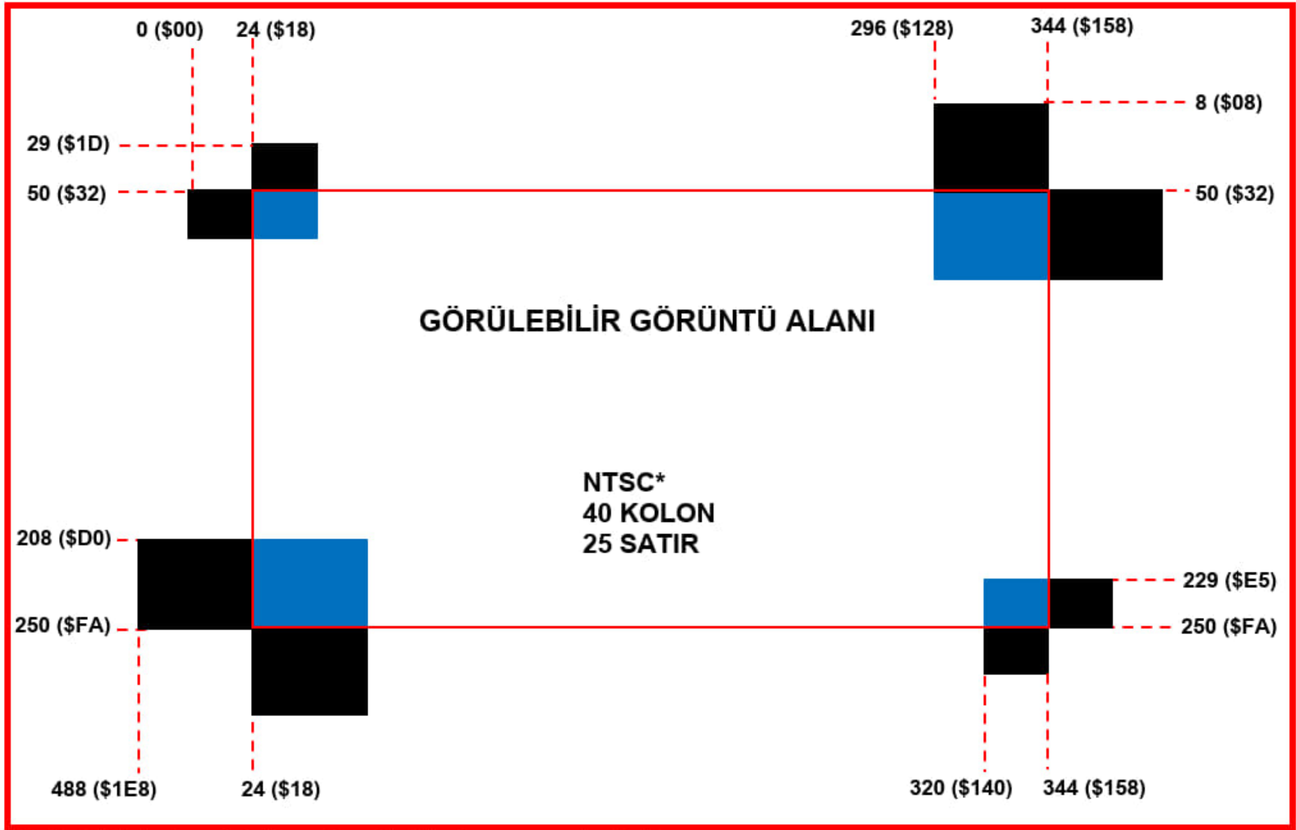
Ekranın üst kısmından itibaren ekran üzerindeki ilk değer ve genişletilmemiş bir yaratık için Y konumu 30'dur. Y yönünde genişletilmiş yaratıklar için ise bu değer 9'dur. Ekranda tamamı görüntülenen genişletilmiş ya da genişletilmemiş yaratığın (21 olası satırın hepsi de görüntülenebilir) ilk Y değeri 50'dir.

Genişletilmemiş bir yaratığın ekranda tamamen görünmesini sağlayan en son Y değeri 229'dur. Bu sayı genişletilmiş yaratıklar için 208 olur. Yaratığın ekrandan tamamen kaybolması için ise, ilk Y değeri 250'dir.

```
10 PRINT "♥" :REM EKRANI TEMIZLER
20 POKE 2040, 13 :REM 0'INCI YARATIK ICI
N VERILERI 13'UNCU BLOKTAN AL
30 FOR I= 0 TO 62: POKE 832+I,129: NEXT
:REM YARATIK 0 VERILERINI BLOK 13'E KOY
40 V = 53248 :REM VIDEO CIPININ BASLANGI
CINI TANIMLA
50 POKE V + 21, 1 :REM YARATIK 0'I GORUN
UR YAP
60 POKE V + 39, 1 :REM YARATIK 0 ICIN RE
NK TANIMLA
70 POKE V + 1, 100 :REM YARATIK 0'IN Y K
ONUMUNU TANIMLA
80 POKE V + 16, 0 : POKE V, 100 :REM YAR
ATIK 0'IN X KONUMUNU TANIMLA
█
```

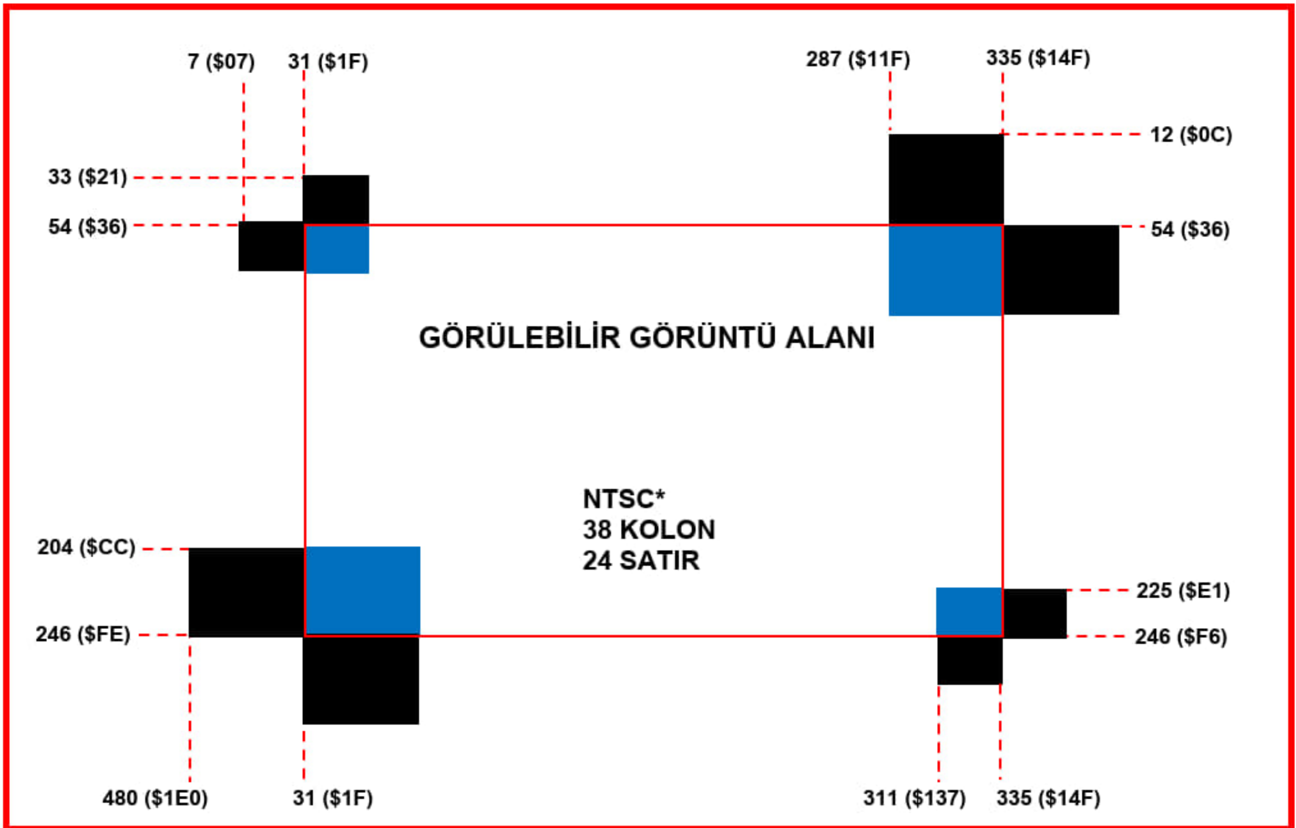
YATAY KONUMLANDIRMA

Yatay yönde konumlama biraz daha karmaşıktır, çünkü bu kez 256'dan daha çok konuma sahipsiniz, Yani X konumunu kontrol eden fazladan bir 9'uncu bitiniz daha var. Gerektiğinde bu biti de ekleyerek yaratığınızın X yönünde, hareket edebileceği 512 olası konum sağlamış olursunuz. Bu ekranda görebileceğimizden çok daha fazla kombinasyonu olanaklı kılar. Her yaratık, 0'dan 511'e kadar olan tüm konumlarda bulunabilir. Bu değerin, yalnızca 24'le 343 arasındakiler, ekranda görülebilirler. Eğer bir yaratığın X konum değeri, 255'ten (ekranın sağ tarafındadır) büyükse, X konumunun en soldaki biti konumu X ESB kaydının değeri 1 olmalıdır. Eğer bir yaratığın X konumu 256'dan küçük ise (ekranın sol tarafında) o zaman da X ESB'nin değeri 0 olmalıdır. X ESB kayıtlarında yer alan bitlerin her biri, bir yaratığa karşılık gelir. [0'ıncı bit 0'ıncı yaratığa, 1'inci bit 1'inci yaratığa, vs.]



* Evinizdeki televizyon için Kuzey Amerika televizyon yayın standarttı.

ŞEKİL 3-3 YARATIK ŞEMASI



* Evinizdeki televizyon için Kuzey Amerika televizyon yayın standarttı.

KONUMLAMA ŞEMASI

Aşağıda verdiğimiz program, yaratığınızın ekrandaki hareketini sağlayacaktır.

```
10 PRINT"♥": REM SHIFT CLR/HOME
20 POKE 2040,13
30 FOR I = 0 TO 62: POKE832+I,129: NEXT
40 V = 53248
50 POKE V + 21,1
60 POKE V + 39,1
70 POKE V + 1, 100
80 FOR J = 0 TO 347
90 HX = INT(J/256): LX = J - 256 * HX
100 POKE V,LX: POKE V + 16,HX: NEXT
```

Genişletilmiş yaratıklar ekranın sol tarafına doğru X yönünde hareket ettiklerinde, SAĞ TARAFTAKİ yaratıkları silmeye başlamalısınız. Çünkü genişletilmiş yaratıklar, ekranın sol tarafında elverişli olan alandan daha büyüktürler.

```
10 PRINT"♥": REM SHIFT CLR/HOME
20 POKE 2040,13
30 FOR I = 0 TO 62: POKE832+I,129: NEXT
40 V = 53248
50 POKE V + 21,1
60 POKE V + 39,1
70 POKE V + 1, 100
80 J = 488
90 HX = INT(J/256): LX = J - 256 * HX
100 POKE V, LX: POKE V + 16,HX
110 J = J + 1: IF J > 511 THEN J = 0
120 IF J > 488 OR J < 348 GOTO 90
```

Şekil 3-3·te yer alan şemalar yaratık konumlandırması için açıklayıcı niteliktedir.

Bu değerleri kullanarak, yaratığınızı istediğiniz yere yerleştirebilirsiniz. Her defasında, 1 noktanın konumunu değiştirmeniz ise size çok yumuşak bir hareket sağlayacaktır.

YARATIK KONUMLANDIRMA ÖZETİ

Normal (genişletilmemiş) yaratıkları kısmen de olsa, en az 40 kolon * 25 satırlık bir modda görmek istiyorsanız aşağıdaki parametreleri kullanmalısınız:

1 < = X < = 343
30 < = Y < = 249

38 kolonluk modda ise X parametreleri şöyle değişir:

8 < = X < = 334

24 satırlık mod için Y parametrelerini yazarsak:

34 < = Y < = 245

Genişletilmiş yaratıklar için ise yine 40 kolon * 25 satır modunda şu parametreler söz konusudur:

489 **>** = X < = 343

9 > = Y < = 249

38 kolon modunda X parametreleri şöyle değişir:

496 > = X < = 334

24 satır modundaki Y parametrelerinin değişimi ise şöyle olur:

13 < = Y < = 245

YARATIK GÖRÜNTÜLEME ÖNCELİKLERİ

Yaratıklar birbirlerinin önlerinden veya arkalarından geçebildikleri gibi birbirlerinin yollarını da kesebilirler. Bu özellik size oyunlarınız için 3 boyutlu bir etki alanı sağlar. Yaratıkların birbirlerine göre öncelikleri vardır. Yani 0 numaralı yaratık 1 numaralı yaratığa göre, 1 numaralı yaratık ise 2 numaralı yaratığa göre öncelik taşır, bu 7 numaralı yaratığa kadar böylece gider. 7 numaralı yaratık en düşük önceliğe sahip olan yaratıktır. Örneklersek: eğer yaratık 1 ve yaratık 6 birbirlerini geçecek şekilde konumlandırılmışsa 1 numaralı yaratık 6 numaralı yaratığın önünde yer alacaktır.

NOT: Bir “pencere” efekti oluşturmak mümkündür. Daha yüksek önceliğe sahip bir hareketli grafiğin içinde "delikler" varsa (noktaların 1'e ayarlanmadığı ve bu nedenle AÇIK duruma geldiği alanlar), daha düşük önceliğe sahip hareketli grafiğin içi görünür. Bu, hareketli grafik ve arka plan verilerinde de olur.

Bu durumda ekranda ön planda görmek istediğiniz yaratıklara vereceğiniz numaralar en düşük numaralar olmalıdır. Ekranın arka bölümünde yer almasını istedikleriniz ise yüksek yaratık numaralarına sahip olanlardır.

Yaratık-zemin önceliği ise 53275 ($D01B) adresindeki yaratık-zemin önceliği kaydı tarafından kontrol edilir. Bu kayıtta her yaratığın bir biti vardır. Eğer bir yaratığa karşılık gelen bit 0 ise bu yaratık, zeminine göre öncelik sahibi demektir. Yani yaratık, zemin verilerinin önünde görünecektir. Eğer aynı bit 1 ise zeminin yaratığa göre önceliği var demektir. Bu durumda yaratık zemin verilerinin ardında görünecektir.

ÇARPIŞMA

VIC-II çipinin en ilginç yönlerinden birisi de çarpışmaları kontrol edebilme yeteneğidir. Çarpışmalar iki yaratık arasında ya da yaratıklarla zemin verileri arasında olabilir.

Bir çarpışmanın olabilmesi için, bir yaratığın 0 olmayan bölümü ile diğerinin (yaratık veya bir karakter) 0 olmayan bölümünün çakışması gereklidir.

YARATIK İLE YARATIK ÇARPIŞMALARI

VIC-II kontrol kaydının 53278 ($D01E) adresinde yer alan yaratığın-yaratıkla çarpışmasını kontrol eden kayıt tarafından tanımlanır. Bu kayıtta, her yaratığın bir biti bulunur. Eğer bu bit 1'se yaratık çarpışabilir. Bu kayıttaki bitler PEEK komutu ile okununcaya kadar oldukları gibi kalırlar. Okunduktan sonra kayıt otomatik olarak temizlenir. Bu yüzden, değeri işiniz bitinceye kadar bir değişkende saklamak daha iyidir.

NOT: Yaratıklar ekranda görünmeseler bile çarpışma olur.

YARATIK İLE VERİ ÇARPIŞMALARI

Yaratıkların verilerle çarpışması VIC-II çipinin kontrol kaydının 53279 ($D01F) adresinde tanımlanır.

Bu kayıtta her yaratığın bir biti bulunur. Eğer bu bit 1 ise yaratıkla veri arasında bir çarpışma söz konusudur. Bu kaydın işleyişi de tıpkı diğeri gibidir.

NOT: Çarpışmalarda çok-renkli veri 01'in saydam olduğu varsayılır. Ekranın zemin rengini değiştirirken her şeyi çok-renkli moddaki 01 ile çarpışmaya neden olmayacak şekilde düzenleyin.

```
10 REM * YARATIKLAR ORNEK 1 *
20 REM UCAN BALON
30 VIC=13*4096:REM VIC KAYITLARININ BASL
ANGIC YERLERI
35 POKEVIC+21,1:REM YARATIK 0'I GORUNEBI
LIR HALE GETIR
36 POKEVIC+33,14:REM ZEMIN RENGINI ACIK
MAVI YAP
37 POKEVIC+23,1:REM 0'INCI YARATIGI Y YO
NUNDE GENISLET
38 POKEVIC+29,1:REM 0'INCI YARATIGI X YO
NUNDE GENISLET
40 POKE2040,192:REM YARATIK 0 GOSTERGECI
NI 1 YAP
180 POKEVIC+0,100:REM YARATIK 0 ICIN X K
ONUMUNU TANIMLA
190 POKEVIC+1,100:REM YARATIK 0 ICIN Y K
ONUMUNU TANIMLA
220 POKEVIC+39,1:REM YARATIK 0'IN RENGIN
I TANIMLA
250 FORY=0TO63:REM YARATIK ICIN KULLANIL
AN DONGUDE BAYT SAYACI
300 READA:REM BIR BAYT OKU
310 POKE192*64+Y,A:REM VERI YERLESTIR
320 NEXTY:REM DONGU TEKRARLA
330 DX=1:DY=1
340 X=PEEK(VIC):REM 0'INCI YARATIGIN X K
ONUMUNA BAKALIM
350 Y=PEEK(VIC+1):REM 0'INCI YARATIGIN Y
 KONUMUNA BAKALIM
```

```
360 IFY=50ORY=208THENDY=-DY:REM EGER Y .
... KENARINDA ISE EKRAN, O ZAMAN DELTA Y
370 REM TERS CEVRILSIN
380 IFX=24AND(PEEK(VIC+16)AND1)=0THENDX=
-DX:REM EGER YARATIK .... SOL KENARA
390 REM DEGDIGINDE (X=24 VE YARATIK 0'IN
MSB'I 0 ISE) GERI DON
400 IFX=40AND(PEEK(VIC+16)AND1)=1THENDX=
-DX: REM EGER YARATIK .... SAG KENARA
410 REM DEGDIGINDE (X=40 VE YARATIK 0'IN
MSB'I 1 ISE) GERI DON
420 IFX=255ANDDX=1THENX=-1:SIDE=1:REM EK
RANIN DIGER TARAFINI AC
440 IFX=0ANDDX=-1THENX=256:SIDE=0:REM EK
RANIN DIGER TARAFINI AC
460 X=X+DX:REM DELTA X'I X'E EKLE
470 X=XAND255:REM X'IN IZIN VERILEN SINI
RLAR ICERISINDE OLDUGUNDAN EMIN OLUN
480 Y=Y+DY:REM DELTA Y'I Y'E EKLE
485 POKEVIC+16,SIDE
490 POKEVIC,X:REM YARATIK 0'IN X KONUMUN
A YENI X DEGERI VERIN
510 POKEVIC+1,Y:REM YARATIK 0'IN Y KONUM
UNA YENI Y DEGERI VERIN
530 GOTO340
600 REM *** YARATIKLAR ICIN VERILER ***
610 DATA 0,127,0,1,255,192,3,255,224,3,2
31,224
620 DATA 7,217,240,7,223,240,7,217,240,3
,231,224
630 DATA 3,255,224,3,255,224,2,255,160,1
,127,64
640 DATA 1,62,64,0,156,128,0,156,128,0,7
3,0,0,73,0
650 DATA 0,62,0,0,62,0,0,62,0,0,28,0,0
```

```
10 REM * YARATIKLAR ORNEK 2 *
20 REM YINE UCAN BALON
30 VIC=13*4096:REM VIC KAYITLARININ BASL
ANGIC YERLERI
35 POKEVIC+21,63:REM NUMARALARI 0'DAN 5'
E KADAR OLAN YARATIKLARI GORUNEBILIR YAP
36 POKEVIC+33,14:REM ZEMIN RENGINI ACIK
MAVI YAP
37 POKEVIC+23,3:REM 0 VE 1'INCI YARATIGI
 Y YONUNDE GENISLET
38 POKEVIC+29,3:REM 0 VE 1'INCI YARATIGI
 X YONUNDE GENISLET
40 POKE2040,192:REM YARATIK 0 GOSTERGECI
NI 1 YAP
50 POKE2041,193:REM YARATIK 1 GOSTERGECI
NI 1 YAP
```

```
60 POKE2042,192:REM YARATIK 2 GOSTERGECI
NI 1 YAP
70 POKE2043,193:REM YARATIK 3 GOSTERGECI
NI 1 YAP
80 POKE2044,192:REM YARATIK 4 GOSTERGECI
NI 1 YAP
90 POKE2045,193:REM YARATIK 5 GOSTERGECI
NI 1 YAP
100 POKEVIC+4,30:REM YARATIK 2 ICIN X KO
NUMUNU TANIMLA
110 POKEVIC+5,58:REM YARATIK 2 ICIN Y KO
NUMUNU TANIMLA
120 POKEVIC+6,65:REM YARATIK 3 ICIN X KO
NUMUNU TANIMLA
130 POKEVIC+7,58:REM YARATIK 3 ICIN Y KO
NUMUNU TANIMLA
140 POKEVIC+8,100:REM YARATIK 4 ICIN X K
ONUMUNU TANIMLA
150 POKEVIC+9,58:REM YARATIK 4 ICIN Y KO
NUMUNU TANIMLA
160 POKEVIC+10,100:REM YARATIK 5 ICIN X
KONUMUNU TANIMLA
170 POKEVIC+11,58:REM YARATIK 5 ICIN X K
ONUMUNU TANIMLA
175 PRINT"[illegible]"TAB(15)"YARATIKLARIMIZ":REM
TERS E CTRL+2, TERS KALP SHIFT+CLR/HOME
176 PRINTTAB(55)"ON TOP OF EACH OTHER"
180 POKEVIC,100:REM YARATIK 0'IN X KONUM
U
190 POKEVIC+1,100:REM YARATIK 0'IN Y KON
UMU
200 POKEVIC+2,100:REM YARATIK 1'IN X KON
UMU
210 POKEVIC+3,100:REM YARATIK 1'IN Y KON
UMU
220 POKEVIC+39,1:REM YARATIK 0'IN RENGI
230 POKEVIC+41,1:REM YARATIK 2'NIN RENGI
240 POKEVIC+43,1:REM YARATIK 4'UN RENGI
250 POKEVIC+40,6:REM YARATIK 1'IN RENGI
260 POKEVIC+42,6:REM YARATIK 3'UN RENGI
270 POKEVIC+44,6:REM YARATIK 5'IN RENGI
280 FORX=192TO193:REM YARATIKLARI TANIML
AYAN DONGUNUN BASLANGICI
290 FORY=0TO63:REM YARATIK ICIN KULLANIL
AN DONGUDE BAYT SAYACI
300 READA:REM BIR BAYT OKU
310 POKEX*64+Y,A:REM VERILERI YARATIK AL
ANINA YERLESTIR
320 NEXTY,X:REM DONGULERI TEKRARLA
330 DX=1:DY=1
340 X=PEEK(VIC):REM 0'INCI YARATIGIN X K
ONUMUNA BAKALIM
350 Y=PEEK(VIC+1):REM 0'INCI YARATIGIN Y
 KONUMUNA BAKALIM
360 IFY=50ORY=208THENDY=-DY:REM EGER Y .
... KENARINDA ISE EKRAN, O ZAMAN DELTA Y
```

```
370 REM TERS CEVRILSIN
380 IFX=24AND(PEEK(VIC+16)AND1)=0THENDX=
-DX:REM EGER YARATIK .... SOL KENARA
390 REM DEGDIGINDE (X=24 VE YARATIK 0'IN
MSB'I 0 ISE) GERI DON
400 IFX=40AND(PEEK(VIC+16)AND1)=1THENDX=
-DX: REM EGER YARATIK .... SAG KENARA
410 REM DEGDIGINDE (X=40 VE YARATIK 0'IN
MSB'I 1 ISE) GERI DON
420 IFX=255ANDDX=1THENX=-1:SIDE=1:REM EK
RANIN DIGER TARAFINI AC
440 IFX=0ANDDX=-1THENX=256:SIDE=0:REM EK
RANIN DIGER TARAFINI AC
460 X=X+DX:REM DELTA X'I X'E EKLE
470 X=XAND255:REM X'IN IZIN VERILEN SINI
RLAR ICERISINDE OLDUGUNDAN EMIN OLUN
480 Y=Y+DY:REM DELTA Y'I Y'E EKLE
485 POKEVIC+16,SIDE
490 POKEVIC,X:REM YARATIK 0'IN X KONUMUN
A YENI X DEGERI VERIN
500 POKEVIC+2,X:REM YARATIK 1'IN X KONUM
UNA YENI X DEGERI VERIN
510 POKEVIC+1,Y:REM YARATIK 0'IN Y KONUM
UNA YENI Y DEGERI VERIN
520 POKEVIC+3,Y:REM YARATIK 1'IN Y KONUM
UNA YENI Y DEGERI VERIN
530 GOTO340
600 REM *** YARATIKLAR ICIN VERILER ***
610 DATA 0,255,0,3,153,192,7,24,224,7,56
,224,14,126,112,14,126,112,14,126,112
620 DATA 6,126,96,7,56,224,7,56,224,7,56
,224,1,56,128,0,153,0,0,90,0,0,56,0
630 DATA 0,56,0,0,0,0,0,0,0,0,126,0,0,42
,0,0,84,0,0,40,0,0
640 DATA 0,0,0,0,102,0,0,231,0,0,195,0,1
,129,128,1,129,128,1,129,128
650 DATA 1,129,128,0,195,0,0,195,0,4,195
,32,2,102,64,2,36,64,1,0,128
660 DATA 1,0,128,0,153,0,0,153,0,0,0,0,0
,84,0,0,42,0,0,20,0,0
```

```
10 REM * YARATIKLAR ORNEK 3 *
20 REM SICAK HAVA G.O.R.F.
30 VIC=53248:REM VIC KAYITLARININ BASL
ANGIC YERLERI
35 POKEVIC+21,1:REM YARATIK 0'I GORUNEBI
LIR HALE GETIR
36 POKEVIC+33,14:REM ZEMIN RENGINI ACIK
MAVI YAP
37 POKEVIC+23,1:REM 0'INCI YARATIGI Y YO
NUNDE GENISLET
38 POKEVIC+29,1:REM 0'INCI YARATIGI X YO
NUNDE GENISLET
```

```
40 POKE2040,192:REM YARATIK 0 GOSTERGECI
NI 1 YAP
50 POKEVIC+28,1:REM COK-RENKLI MODA GEC
60 POKEVIC+37,7:REM COK-RENKLI 0'I BIR Y
AP
70 POKEVIC+38,4:REM COK-RENKLI 1'I BIR Y
AP
180 POKEVIC+0,100:REM YARATIK 0 ICIN X K
ONUMUNU TANIMLA
190 POKEVIC+1,100:REM YARATIK 0 ICIN Y K
ONUMUNU TANIMLA
220 POKEVIC+39,2:REM YARATIK 0'IN RENGIN
I TANIMLA
290 FORY=0TO63:REM YARATIK ICIN KULLANIL
AN DONGUDE BAYT SAYACI
300 READA:REM BIR BAYT OKU
310 POKE12288+Y,A:REM VERILERI YARATIK A
LANINA YERLESTIR
320 NEXTY:REM DONGUYU TEKRARLA
330 DX=1:DY=1
340 X=PEEK(VIC):REM 0'INCI YARATIGIN X K
ONUMUNA BAKALIM
350 Y=PEEK(VIC+1):REM 0'INCI YARATIGIN Y
 KONUMUNA BAKALIM
360 IFY=50ORY=208THENDY=-DY:REM EGER Y .
... KENARINDA ISE EKRAN, O ZAMAN DELTA Y
370 REM TERS CEVRILSIN
380 IFX=24AND(PEEK(VIC+16)AND1)=0THENDX=
-DX:REM EGER YARATIK .... SOL KENARA
390 REM DEGDIGINDE (X=24 VE YARATIK 0'IN
MSB'I 0 ISE) GERI DON
400 IFX=40AND(PEEK(VIC+16)AND1)=1THENDX=
-DX:REM EGER YARATIK .... SAG KENARA
410 REM DEGDIGINDE (X=40 VE YARATIK 0'IN
MSB'I 1 ISE) GERI DON
420 IFX=255ANDDX=1THENX=-1:SIDE=1:REM EK
RANIN DIGER TARAFINI AC
440 IFX=0ANDDX=-1THENX=256:SIDE=0:REM EK
RANIN DIGER TARAFINI AC
460 X=X+DX:REM DELTA X'I X'E EKLE
470 X=XAND255:REM X'IN IZIN VERILEN SINI
RLAR ICERISINDE OLDUGUNDAN EMIN OLUN
480 Y=Y+DY:REM DELTA Y'I Y'E EKLE
485 POKEVIC+16,SIDE
490 POKEVIC,X:REM YARATIK 0'IN X KONUMUN
A YENI X DEGERI VERIN
510 POKEVIC+1,Y:REM YARATIK 0'IN Y KONUM
UNA YENI Y DEGERI VERIN
520 GETA$:REM KLAVYEDEN BIR TUSA BASIN
521 IFA$="M"THENPOKEVIC+28,1:REM KULLANI
CI COK-RENKLI MODU SECTI
522 IFA$="H"THENPOKEVIC+28,0:REM KULLANI
CI YUKSEK-COZUNURLUKLU MODU SECTI
530 GOTO340
600 REM *** YARATIKLAR ICIN VERILER ***
```

```
610 DATA 64,0,1,16,170,4,6,170,144,10,17
0,160,42,170,168,41,105,104,169,235,106
620 DATA 169,235,106,169,235,106,170,170
,170,170,170,170,170,170,170,170,170,170
630 DATA 166,170,154,169,85,106,170,85,1
70,42,170,168,10,170,160,1,0,64,1,0,64
640 DATA 5,0,80,0
```

DİĞER GRAFİK ÖZELLİKLERİ

EKRAN TEMİZLEME

VIC-11 kontrol kaydının 4 'üncü biti ekranın temizlenmesi işlevini kontrol eder. Bu bit 53265 ($D011) adresindeki kontrol kaydında yer alır. Bu bitin değeri 1 ise ekran normaldir. 4'uncü biti 0'a eşitlerseniz tüm ekranın çerçeve rengini aldığını görürsünüz.

Aşağıdaki POKE yönergesi tüm ekranı temizleyecektir. Bu, verilerin kaybolması anlamına gelmez, yapılan sadece verilerin görüntülenmesini önlemektir.

```
POKE 53265,PEEK(53265)AND 239
```

Ekranda sildiklerinizi tekrar görmek istiyorsanız kullanmanız gereken POKE komutu ise;

```
POKE 53265, PEEK (53265) OR 16
```

NOT: Ekranı temizlemeniz işlemciyi hızlandıracak, böylece o anda işlemekte olan program da hızlanacaktır.

TARAMA KAYDI

Tarama kaydı, VIC-II çipinin 53266 ($D012) adresinde yer alır ve çift amaçlı bir kayıttır. Bu kaydı okuduğunuzda, o anki tarama konumunun en alt 8 bitini elde edersiniz. En soldaki bitin tarama konumu, 53265 ($D011) adresindeki kayıttadır. Tarama kaydı, ekrandaki titremeleri önlemek için görüntüdeki zamanlama değişimlerini düzenlemekte kullanılır.

KESİNTİ DURUM KAYDI

Bu kaydın kullanım amacı, kesinti kaynağının, o andaki durumunu göstermektir. İki yaratık çarpıştıkları anda, kesinti kaydının 2'inci biti 1 olur. Bu durum, aşağıdaki tabloda yer alan 0-3 bitleri için bire bir ilişkide de geçerlidir. Kesinti olduğu anda 7'inci bit de 1 değerini alır.

Kesinti durum kaydı 53273 ($D019) adresinde yer alır ve aşağıdaki gibidir:

LATCH	BİT NO.	AÇIKLAMA
IRST	0	O anki tarama sayıldığında 1 olur = saklanmış tarama sayımı.
IMDC	1	YARATIK-VERİ çarpışması olduğunda set edilir (Reset oluncaya kadar, yalnızca 1'inci bit.)
IMMC	2	YARATIK-YARATIK çarpışması olduğunda set edilir (Reset'e kadar yalnızca 1'inci bit.)
ILP	3	Light-pen'in negatif geçişi ile set edilir (Her çerçevede 1 defa.)
IRQ	4	Latch set edildiğinde set edilir ve işleme sokulur.

Kesinti biti set edildiğinde, "mandallanır" ve kesintiyi işlemek için hazır olduğunuzda, kesinti kaydındaki bu bitin, 1 yazılarak temizlenmesi gerekir. Bu, diğer kesinti bitlerinin saklanmasına gerek duyulmaksızın istediğiniz kesinti bitlerinin işlenmesine olanak sağlar. **KESİNTİ ETKİNLEŞTİRME KAYDI** 53274 ($D01A) adresindedir ve formatı kesinti durum kaydının formatı ile aynıdır. Kesinti etkinleştirme kaydındaki uygun bit 1 yapılmadıkça, bu kaynaktan gelen herhangi bir kesinti değerlendirilemez.

Bir kesinti isteğinin kabul edilmesi için, uygun olan kesinti etkinleştirme bitine (yukarıdaki tabloda gösterilen) 1 yerleştirilmesi gerekir.

Bu güçlü kesinti yapısı, ekranın bölünmesine yarayan modların kullanılabilmesini sağlar. Örneğin, ekranın yarısını bit haritalanmış, yarısını metin olarak ya da bir kerede 8 yaratıktan fazlasını, vs. elde etmenizi sağlar, bu işin sırrı, kesintilerin doğru dürüst kullanılmasındadır. Örneğin; ekranın üst kısmındaki yarının bit haritalı, alt kısmının ise metin olmasını isterseniz, tarama kıyaslama kaydını (daha önce anlatılmıştı) ekranın alt yarısı için tanımlamalısınız. Kesinti olduğunda, VIC-II çipine karakterleri ROM'dan almasını söyleyin ve sonra tarama kıyaslama kaydını ekranın üst kısmındaki kesinti için düzenleyin. Kesinti ekranın üst kısmında oluştuğunda VIC-II çipine, karakterleri RAM'dan almasını söyleyin (Bit harita modu). Karakterlerin ROM'dan ve RAM'dan alınmasını daha önceden anlatılmıştı.

Aynı yolla, 8 yaratıktan fazlasının görüntülenmesini sağlayabilirsiniz. Fakat bunu iyi şekilde yapması için, BASIC'in yeterli ölçüde hızlı olmasını beklemeyin. Bu yüzden işe görüntü kesintilerini kullanarak başlamak isterseniz, makina dilini kullanın.

ÖNERİLEN EKRAN VE KARAKTER RENK KOMBİNASYONLARI

Renkli TV setlerinin belirli renkleri aynı çizgide koyan yan yana gösterme özellikleri sınırlıdır. Aşağıdaki tablo, ekran ve karakter renklerinin hangi kombinasyonlarından kaçınmamız gerektiğini belirtirken hangi renklerle iyi bir uyum elde edebileceğinizi de göstermektedir.

KARAKTER RENGİ (sütunlar) / **EKRAN RENGİ** (satırlar)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	x	☻	x	☻	☻	☺	x	☻	☻	x	☻	☻	☻	☻	☻	☻
1	☻	x	☻	x	☻	☻	☻	x	☺	☻	☺	☻	☻	x	☻	☻
2	x	☻	x	x	☺	x	x	☻	☻	x	☻	x	x	x	x	☺
3	☻	x	x	x	x	☺	☻	x	x	x	x	☺	x	x	☺	x
4	☻	☺	x	x	x	x	x	x	x	x	x	x	x	x	x	☺
5	☻	☺	x	☺	x	x	x	x	x	x	x	☺	x	☻	x	☺
6	☺	☻	x	☻	x	x	x	x	x	x	x	x	x	☺	☻	☻
7	☻	x	☻	x	x	x	☺	x	☺	☻	☺	☻	☻	x	x	x
8	☺	☻	☻	x	x	x	x	☻	x	☻	x	x	x	x	x	☺
9	x	☻	x	x	x	x	x	☻	☻	x	☻	x	x	x	x	☻
10	☺	☺	☻	x	x	x	x	☺	x	☻	x	x	x	x	x	☺
11	☻	☻	x	☺	x	x	x	☻	x	x	x	x	☻	☻	☺	☻
12	☻	☻	☺	x	x	x	☺	x	x	☺	x	☻	x	x	x	☻
13	☻	x	x	x	x	☻	☺	x	x	x	x	☻	x	x	x	x
14	☻	☻	x	☻	x	x	☻	x	x	x	x	☺	x	x	x	☺
15	☻	☻	☻	x	☺	☺	☻	x	x	☺	☺	☻	☻	x	☺	x

☻ **MÜKEMMEL**
☺ **BİRAZ İYİ**
x **KÖTÜ**

YARATIK PROGRAMLAMA-BİR BAŞKA BAKIŞ

Eğer grafikler konusunda güçlük çekiyorsanız, bu bölümü okuyabilirsiniz. Bu bölüm, yaratıklar konusunda çok daha eğitici bir yaklaşım gözetilerek hazırlanmıştır.

BASIC'LE YARATIK OLUŞTURABİLİRSİNİZ-KISA BİR PROGRAM

Commodore 64'te, grafik ve çizgi animasyonları için kullanılabilecek en az 3 programlama tekniği vardır. Bunun için bilgisayarınızda hazır bulunan karakter setini kullanabileceğiniz gibi, karakterlerinizi kendiniz de programlayabilirsiniz. Bunun nasıl yapılacağını aşağıdaki verilecek örnekte satır satır açıklanarak ayrıntılı bir şekilde size sunulacak. Örnekteki bazı değerleri değiştirerek sizde farklı sonuçlar elde edebilirsiniz.

Dahası, bilgisayarınızda hazır bulunan "yaratık grafiklerini" kullanabilirsiniz. Aşağıda verdiğimiz program örneği bu amaçla kullanabileceğiniz en kısa BASIC dili programlarından birisidir:

```
10 PRINT"♥": REM SHIFT+CLR/HOME
20 POKE2040, 13
30 FORS=832TO832+62:POKES,255:NEXT
40 V=53248
50 POKEV+21,1
60 POKEV+39,1
70 POKEV,24
80 POKEV+1,100
```

Bu program herhangi bir yaratığı oluştururken gereksineceğiniz tüm anahtar "bileşenleri" kapsamaktadır. POKE komutu ile kullanacağınız sayıları daha sonra vereceğimiz YARATIK OLUŞTURMA tablosundan bulabilirsiniz. Bu programla elde edeceğiniz ilk yaratık (0 numaralı yaratık) beyaz bir kare biçiminde olacaktır. Şimdi tüm programı satır satır açıklayalım.

SATIR 10, ekranı temizler.

SATIR 20, yaratık göstergecine Commodore 64'ünüzün yaratık verilerini okuyacağı baytları yerleştirir. Yaratık-0 2040'a, yaratık-1 2041'e ve nihayet yaratık-7 2047'ye yerleştirilir. 20 no.'lu satırın yerine şu satırı tekrar girin: 20 FOR SP = 2040TO2047: POKE SP, 13: NEXT SP

Böylece yaratık-göstergeçlerinin 8'ine de 13 yerleştirmiş olacaksınız.

SATIR 30, yaratık-0'ı, RAM belleğinin 832"inci yerleşiminden başlayarak 63 baytlık bir bölümüne yerleştirir (her yaratık için gereken bayt sayısı 63'tür). Böylece ilk yaratık, 832-894 arasındaki yerleşimlerde adreslenmiş olur.

SATIR 40, Video çipinin başlangıç adresini tanımlayan V değişkenini 53248 değerine eşitler. Bu satır yaratıkları yerleştirirken (V+sayı) biçimini kullanmamızı sağlayacaktır. Yaratıkları yerleştirirken (V+sayı) şeklini kullanıyoruz, çünkü bu daha az bellek kullanmamızı sağlayacağı gibi, daha küçük sayılarla çalışmamıza da olanak sağlayacaktır. Örneğin 50'inci satırdaki yönergede, POKE V+21 yazdık. Bu, bizi POKE 53248 + 21 veya POKE 53269 yazmamızla aynı sonuca götürür. Ancak V + 21 daha az yer kaplayacak ve hatırlamak her zaman daha kolay olacaktır.

SATIR 50, yaratık-0'ı harekete geçirir. 0'dan 7'ye kadar numaralanmış 8 yaratık vardır. Bir yaratığı veya birkaçını birlikte harekete geçirmek istiyorsanız, bütün yapacağınız POKE V + 21'den sonra 0'la 255 arasında bir sayı yazmaktır. 0 tüm yaratıkları kapatır (OFF). 255 ile ise tümü açık (ON) durumdadır. Aşağıdaki sayıları kullanarak (POKE) bir veya birkaç yaratığı hareketli duruma getirebilirsiniz.

HEPSİ AÇIK	YARATIK 0	YARATIK 1	YARATIK 2	YARATIK 3	YARATIK 4	YARATIK 5	YARATIK 6	YARATIK 7	HEPSİ KAPALI
V+21,255	V+21,1	V+21,2	V+21,4	V+21,8	V+21,16	V+21,32	V+21,64	V+21,128	V+21,0

POKE V+21,1 yaratık-0'ı, POKE V+21,128 ise yaratık-7'yi açar. Yaratıkların farklı birleşimlerini oluşturmanız da mümkündür. Örneğin POKE V+21,129 hem yaratık-0'ı hem de yaratık 7'yi harekete geçirir. (İki yaratığın "AÇIK" numaraları toplanarak böyle sonuçlar elde etmek mümkündür.)

SATIR 60, yaratık-0'ın rengini belirler. 0'la 15 arasında numaralandırılmış, olası 16 renkten birini seçebilirsiniz. (0: siyah, 15: gri). Her yaratığın rengi ayrı bir POKE yönergesi girilerek sağlanır. Bu yönergelerde V+39 ile V+46 arası kullanılır. POKE V+39,1 yönergesi, yaratık-0'ın rengini beyaz olarak belirler. POKE V+46,15 ise yaratık-7'nin gri olmasını sağlayan yönergedir.

Yarattığınız karakter, siz bilgisayarınızı kapatıncaya kadar bellekte kalır. Böylece istediğiniz yaratığın rengini, konumunu hatta biçimini değiştirebilmeniz mümkün olmaktadır. Örneğin: şimdi yukarıdaki programı çalıştırın (RUN) ve doğrudan modda POKE V+39,8 (satır numarası vermeksizin) girerek **RETURN** tuşuna basın.

Ekrandaki yaratığın rengi artık turuncudur. 0-15 arasındaki sayıları deneyerek, daha birçok renk elde edebilirsiniz. Şimdi programınızı tekrar çalıştırın (RUN). Yaratığınızın tekrar beyaz olduğunu göreceksiniz.

SATIR 70, yaratığın ekrandaki yatay (X) konumunu belirler. Bu sayı yaratığın SOL ÜST KÖŞE hanesine karşılık gelir. Yaratığı ekran dışına, 0 konum noktasına değin hareket ettirebilirsiniz. Ancak, ekranda görebileceğiniz en son konum noktası her zaman 24'tür.

SATIR 80, yaratığın ekrandaki dikey (Y) konumunu belirler. Bu programda, biz, yaratığın X konumunu 24, Y konumunu ise 100 olarak belirledik. POKE V,24: POKE V+1,50 yönergesini doğrudan modda girerseniz (**RETURN** tuşuna basmayı unutmayın), farklı bir yerleşim elde etmiş olacaksınız.

Bu yönerge, yaratığı, ekranın sol üst köşesine yerleştirir. Yaratığı daha aşağıya çekmek için POKE V,24: POKE V+1,229 yazın:

Yaratık-0'ın adresinde yer alan 832'den 895'e kadar her sayı, 8 piksellik bir bloğu gösterir. Bir yaratığın her yatay sırasında, 3 tane 8 piksellik blok bulunur.

80'inci satır tarafından tanımlanan döngü (LOOP), bilgisayarın ilk 8 pikselinin içinin dolmasını sağlar (POKE 832,255). Daha sonra gelen POKE 833,255 yönergesi ise, ikinci 8 piksellik bloğun dolmasını sağlar. Bu işleyiş, sağ alt köşedeki yaratıkta yer alan son piksel grubunun dolmasına değin sürer. Bu işleyişi daha iyi görmek istiyorsanız, şu POKE 833,0 yönergeyi direkt modda girin:

İkinci 8'lik piksel grubunun silindiğini göreceksiniz. Geri dönüşü, POKE 833,255 yönergesini yazarak ya da programınızı tekrar çalıştırarak (RUN) sağlayabilirsiniz.

Aşağıda verilen satırı girmeniz ise, yaratıklarınızın tam ortasındaki blokların silinmesine neden olacaktır.

```
90 FOR A=836 TO 891 STEP3: POKE A,0:
NEXT A
```

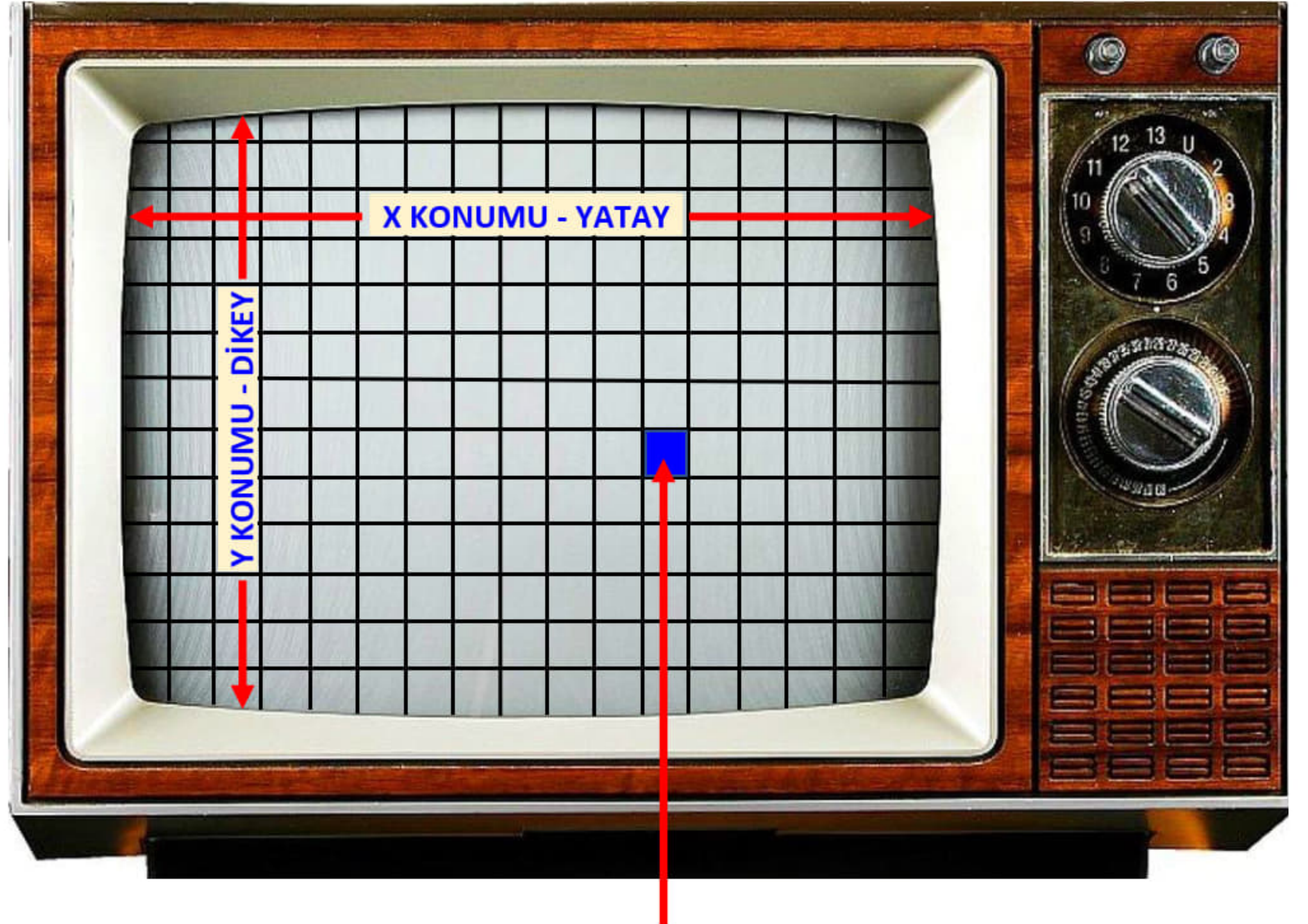
Yaratıkları oluşturan piksellerin, 8'lik bloklardan oluştuğunu hatırlayacaksınız. Bu satır, 8'lik piksel gruplarından 5'inci grubu (blok 836) ve 890'ıncı bloğa kadar tüm üçüncü blokları silecektir. 832'den 894'e kadar olan tüm sayıları kullanabilirsiniz. Bu sayıları 255 ile birlikte kullanırsanız yaratıklarınızın içini dolduracak. 0 ile birlikte kullanırsanız, boş bırakmış olacaksınız.

YARATIK PROGRAMLARINIZI KISALTABİLİRSİNİZ

Yukarıda verdiğimiz program oldukça küçüktü. Ancak bu programı daha da kısaltmanız mümkündür. Biz örneğimizde, neler olup bittiğini iyice anlayabilmeniz açısından, tüm anahtar yerleştirmeleri ayrı satır olarak verdik. Oysa bu program iyi bir programcı için iki satırda yazılacaktır. Şimdi bu iki satır programı görelim.

```
10 PRINTCHR$(147):V=53248:POKEV+21,1:POK
E2040,13:POKEV+39,1
20 FORS=832TO894:POKES,255:NEXT:POKEV,24
:POKEV+1,100
```

TV EKRANI



Ekranın herhangi bir noktasına yerleştirilmiş bir yaratığın hem X, hem de Y konumunu tanımlamalısınız. Ancak bu şekilde onu görüntüleyebilirsiniz.

Şekil 3-4. Görüntü ekranı X ve Y koordinatları olarak bölünmüştür.

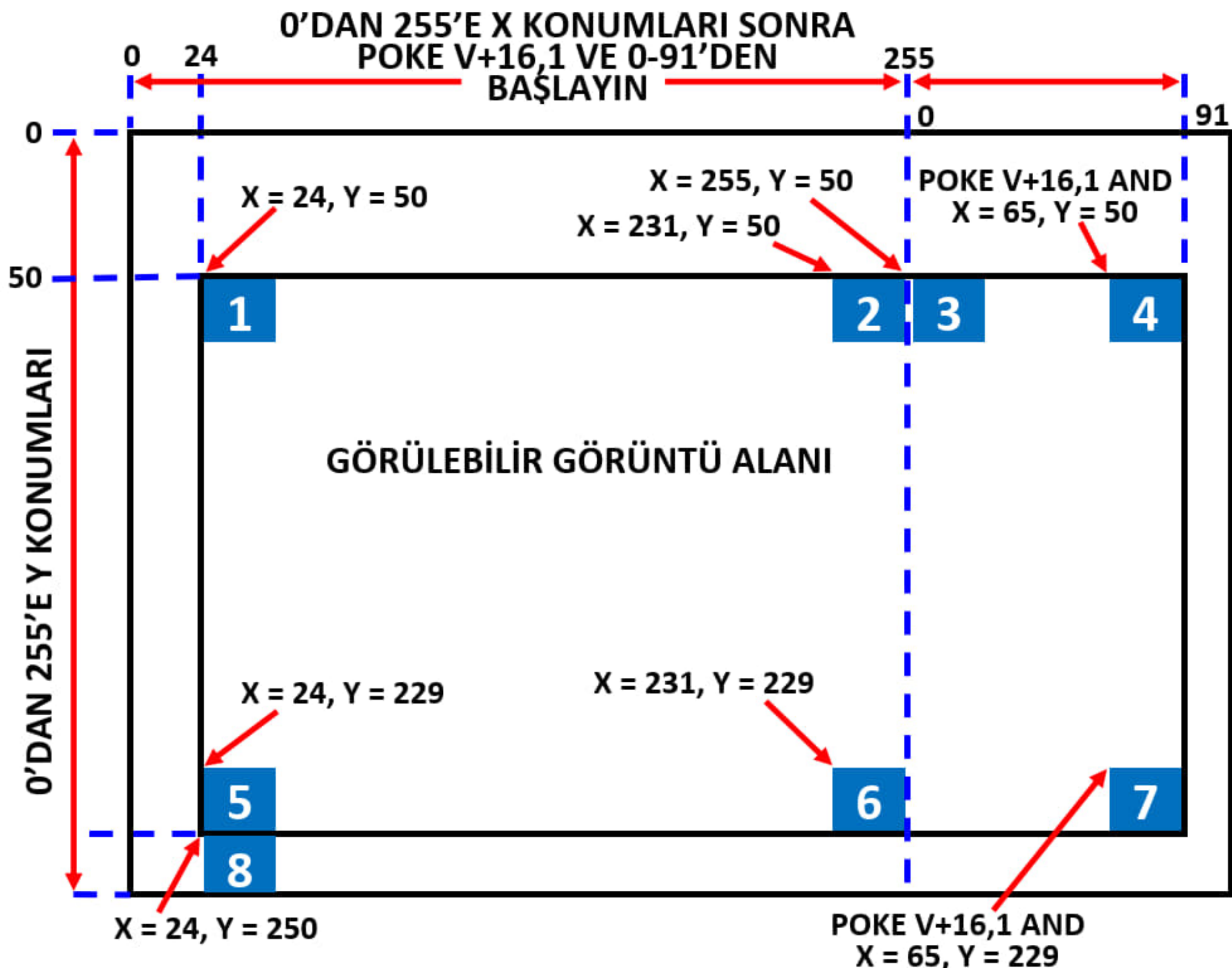
YARATIKLARIN EKRAN ÜZERİNDE YERLEŞTİRİLMESİ

Ekran, tıpkı grafik çizimlerinde olduğu gibi X ve Y koordinatlarına bölünmüştür. X koordinatı YATAY konumlama, Y koordinatı ise dikey konumlama için kullanılır. (Bkz: Şekil 3-4)

Bir yaratığı ekranda görüntülemek istiyorsanız iki yönlü bir konumlama yapmanız şarttır. Yani iki koordinat içinde POKE işlemi yapmalısınız: X konumu ve Y konumu. Bunlar bilgisayarınıza, yaratığınızın üst sol köşesini nerede görüneceğini söyler.

Bir yaratığın 504 pikselden oluştuğunu anımsayacaksınız. (24 satır*21 sütun). Bu durumda ekranınızın sol üst köşesine yerleştireceğiniz bir yaratık, tanımladığınız X-Y konumundan başlayarak 24 yatay piksel, 21 düşey pikselden oluşan bir nesne olarak görüntülenecektir. 24 x 21 piksellik yaratık alanının ne kadarlık bölümünü kullanırsanız kullanın, görüntülemede her zaman tanımlanan yaratığın sol üst köşesi temel alınır.

Bu yaratığın X-Y konumlamasının nasıl işlediğini anlamak istiyorsanız, aşağıdaki şekli incelemenizi öneririz. X-Y numaraları görüntü ekranınızla bağlantılı olarak verilmiştir. Gri olarak gösterilen bölge, televizyonunuzun görüntü alanıdır. Beyaz olarak gösterilen bölge ise sizin görüş alanınız dışında olan konum noktalarını gösterir.



Şekil 3·5. X-Y yaratık konumlarını belirleme.

Bir yaratığ1 istediğiniz yerde görüntülemeniz için yaratığın yerini tanımlayacak olan X ve Y konumlarını yerleştirmeniz şarttır. Bu arada bu yaratığın ayrı X ve Y POKE komutlarına sahip olduğunu unutmayın. Aşağıda 8 yaratığın her biri için, ayrı ayrı X ve Y yerleşimleri verilmiştir:

X VE Y KONUMLARI İÇİN POKE KOMUTLARINDA AŞAĞIDAKİ DEĞERLERİ KULLANIN

	YARATIK 0	YARATIK 1	YARATIK 2	YARATIK 3	YARATIK 4	YARATIK 5	YARATIK 6	YARATIK 7
X Y SAĞ X	V,X V+1,Y V+16,1	V+2,X V+3,Y V+16,2	V+4,X V+5,Y V+16,4	V+6,X V+7,Y V+16,8	V+8,X V+9,Y V+16,16	V+10,X V+11,Y V+16,32	V+12,X V+13,Y V+16,64	V+14,X V+15,Y V+16,128

X KONUMUNUN POKE İŞLEMİ: X konumu için 0'la 255 arasındaki tüm değerleri, soldan sağa doğru sayarak kullanabilirsiniz. 0'la 23 arasında vereceğiniz her değer, yaratığın bir bölümünün ya da tümünün görüş alanınızın dışında kalmasına sebep olacaktır. 24'ten 255'e kadar verilen her değer ise yaratığın görüş alanında olmasını sağlar (255'inci X konumundan sonraki değerler için bir sonraki paragrafa bakınız). Yaratığınızı bu konumlardan birine yerleştirmek için tek yapacağınız X konumunu tanımlayan bir POKE yönergesi yazmaktır. Örneğin: Yaratık 1'i, X konumunun görülen bölgesinin en soluna yerleştirmek istediğinizi düşünelim. Şu yönergeyi yazmalısınız: (POKE V+ 2,24).

255'TEN BÜYÜK X DEĞERLERİ: Ekranda 255'ten büyük konumları kullanmak için ikinci bir POKE yönergesi yazmanız gerekir. Bunun için Şekil 3-5'te yer alan SAĞ X sayılarını kullanabilirsiniz. Normalinde, eğer kayıtlar sadece 8 bit içermeseydi yatay (X) kullanabileceğiniz sayılar 255, 256, 257, vs. olabilirdi. Ancak şimdi ekranın SAĞ yanına erişebilmek için ikinci bir kayıt daha kullanmak zorundasınız. Bu durum için POKE V+16 ile birlikte bir sayı kullanacaksınız (bu sayı yaratık sayısına bağlıdır). Böylece ekranın görülebilir alanı içinde SAĞ'da yer alan fazladan 65 X konumuna daha sahip olacaksınız.

Y KONUMUNUN POKE İŞLEMİ: Y konumu için 0'la 255 arasındaki tüm değerleri kullanabilirsiniz. Bu kez yukarıdan aşağı doğru sayacaksınız. 0'la 49 arasında vereceğiniz değerler yaratığın bir kısmının ya da tümünün ekranın üst kısmında, görüş alanının dışında kalmasına neden olacaktır. 50 ile 229 arasındaki değerler ise yaratığı, görüş alanı içine yerleştirir. Yine 230 ile 255 arasındaki değerler, yaratığı bütünüyle ya da kısmen ekranın alt kısmında, görüş alanı dışında bırakan değerlerdir.

Şimdi X-Y konumlamasının nasıl çalıştığını görelim. Bunun için Yaratık-1'i kullanacağız. Şu programı yazın:

```
10 PRINT"♥":V=53248:POKEV+21,2:POKE2041,
13:FORS=832TO895:POKES,255:NEXT
20 POKEV+40,7
30 POKEV+2,24
40 POKEV+3,50
```

Bu basit programla yaratık-1'i ekranın sol üst köşesinde yer alan içi dolu bir kutu olarak göstereceksiniz. Şimdi 40 numaralı satırı değiştirin:

```
40 POKEV+3,229
```

Bu yönerge yaratığın, ekranda sol alt köşeye kaymasına neden olacaktır. Şimdi bu yaratığın sağ X limitini kontrol edelim. 30 numaralı satırın yerine şu satırı girin:

```
30 POKEV+2,255
```

Bu yönerge yaratığın sağa kaymasına yol açar. Ancak bu kez yaratık sağ X limitine, yani 255'e yerleştirilmiştir. Bu noktada 16 no.'lu kayıtta bulunan en soldaki biti ayarlamak gerekir. Bir başka söyleyişle; POKE V+16 yazarak bu yönergeyi, X-Y POKE çizelgesindeki "sağ X" numaralarından biriyle tamamlayın. Bu yönerge X konum sayacının ekrandaki 256'ıncı pikselden başlayarak yeniden işlemesini sağlayacaktır. 30 numaralı satırı aşağıdaki yönerge ile değiştirin:

```
30 POKEV+16,PEEK(V+16) OR 2:POKE V+2.0
```

POKE V+16,2; X konumunun en soldaki bitine yaratık 1'in yerleşmesini ve ekranın 256'ıncı piksel/konumundan yeniden başlamasını sağlar. POKE V+2,0 yönergesi ise yaratığın, artık 256'ıncı piksele karşılık gelen yeni konum sıfırda görüntülenmesini sağlayacaktır.

Ekranın sol yanına geri dönmek istiyorsanız: X konumunun en soldaki bitine 0 değerini vermelisiniz. Şu yönerge bunu sağlayacaktır:

```
30 POKEV+16,PEEK(V+16) AND 253
```

Şimdi X yönünde konumlamanın aşamalarını özetleyelim.

Söz konusu yaratık için bir X konumu yerleştirin (POKE ile). Bu yönerge için kullanacağınız sayı 0'la 255 arasında olmalıdır. 255 konum/pikselinden öte konumlar için, fazladan bir POKE(V+16) yönergesi daha kullanmanız gerekir. Bu yönergeyle X konumunu tekrar 0'dan; ama bu kez 256'ıncı pikselden başlayarak sayacaksınız.

(Örnek: POKE V+16, PEEK(V+16) AND 254 yönergesini kullanabilirsiniz.)

BİRDEN FAZLA YARATIĞIN EKRAN KONUMLAMASI

Aşağıdaki verdiğimiz program, 3 farklı yaratığı (0, 1 ve 2) değişik renklerle, değişik konumlara yerleştirmenizi sağlayacaktır.

```
10 PRINT"♥":V=53248
20 FORS=832TO895:POKES,255:NEXT
30 FORM=2040TO2042:POKEM,13:NEXT
40 POKEV+21,7:POKEV+39,1:POKEV+40,7
50 POKEV+41,8:POKEV,24:POKEV+1,50
60 POKEV+2,12:POKEV+3,229
70 POKEV+4,255:POKEV+5,50
```

Burada. 3 yaratığın 3'ü de, verilerini aynı yerden alan kareler olarak tanımlanmıştır. Önemli olan nokta; bu 3 yaratığın nasıl konumlandırılacağıdır. 0 numaralı beyaz yaratık, ekranın sol üst köşesine, 1 numaralı yaratık ise ekranın sol alt köşesine ancak yarısı görülebilecek şekilde yerleştirilmiştir. (24'ün görüş alanı içerisindeki en sol konum olduğunu hatırlayın. 24'ten küçük olan tüm X konum numaraları yaratığınız, ekranın dışına çeker). Biz burada X konumu olarak 12'yi verdik. Böylece de yaratığın yarısının ekranın görülebilir alanı dışında görüntülenmesini sağladık. Ve son olarak da 2 numaralı turuncu yaratık sağ X limite yerleştirilmiş oldu. Şimdi geriye yanıtlamamız gereken 1 soru kaldı: Eğer yaratığı 255'inci X konumunun sağında görüntülemek isterseniz, ne yapacaksınız?

YARATIĞIN 255'İNCİ X KONUMUNUN ÖTESİNDE GÖRÜNTÜLENMESİ

Bir yaratığı 255'inci X konumunun ötesinde görüntülemek için özel bir POKE yönergesi kullanmanız gerekir. Bu yönerge ekran boyunca bu kez 256'ıncı piksel konumundan başlayarak sayabilmenizi sağlayacaktır. Şimdi olayın nasıl işlediğine bir bakalım.

Önce söz konusu yaratık için belirlenmiş bir sayı ile birlikte POKE V+16 yönergesini girin. (X-Y şemasının sağ X sırasını kontrol etmelisiniz. Biz yaratık 0'ı kullanacağız). Şimdi bir X konumu tanımlayalım. Bu tanımı yaparken X sayacının, ekranın 256'ıncı konumunu 0 olarak tanımlayacağını unutmayın. 50 numaralı satırı aşağıdaki yönergeyle değiştirin:

```
50 POKEV+16,1:POKEV,24:POKEV+1,75
```

Bu satır V+16'ya ekranın sağ tarafını açacak bir sayı yerleştirecektir. Bu örnekte yaratık-0 için geçerli olan yeni 24'üncü X konumu artık 255'inci X konumunun 24 piksel daha sağından başlayacaktır. Ekranın sağ tarafını, 60 numaralı yönergeyi, aşağıdaki yönergeyle değiştirerek kontrol edebilirsiniz.

```
50 POKEV+16,1:POKEV,65:POKEV+1,75
```

Yaratık şemasında verilen değerlerle yapacağınız birkaç denemeden sonra yaratıklarınızı istediğiniz konuma yerleştirmeniz ve hareketlendirmeniz için gerekli olan değerleri bulacaksınız. Ayrıca "Yaratıkların hareketlendirilmesinden söz eden bölüm, bu konudaki çelişkilerinizi çözmenize de yardımcı olacaktır.

YARATIK ÖNCELİKLERİ

Ekranınızda: birbirinin önünde ya da arkasında hareket eden farklı yaratıklar elde etmeniz mümkündür. Bu inanılmaz 3 boyutlu görüntü, bilgisayarınızda hazır bulunan yaratık öncelikleri tarafından gerçekleştirilir. Burada yapılan, ekranda iki ya da daha fazla yaratığın çakışması (üst üste gelmesi) koşulunda hangi yaratığa diğerlerine göre öncelik tanınacağının belirlenmesidir.

Yerleşik kuralımız burada da geçerlidir: ilk giren, ilk çıkar" (FIFO). Bunun anlamı küçük sayılı yaratıkların diğerlerine göre otomatik olarak öncelik taşıyacaklarıdır. Örneğin; Yaratık-0 ve Yaratık-1'i aynı noktada ekranda görüntülemek isterseniz, iki yaratık çakışacaktır. Bu durumda Yaratık-0, yaratık-1'in önünde görünecektir. Zaten yaratık-0 her zaman diğer yaratıklara göre öncelikli olacaktır. Çünkü en küçük numaraya sahip yaratıktır. Bir karşılaştırma yaparsak yaratık-1, yaratık-2 ve 7'ye göre önceliklidir. Yaratık-7 (son yaratık) ise diğer tüm yaratıklara göre en az önceliği olan yaratıktır ve çakışma durumunda her zaman diğer yaratıkların arkasında kalacaktır. Öncelik işleyişini göstermek için yukarıda verdiğimiz programın 50. 60. ve 70'inci satırlarını aşağıdaki gibi değiştirelim:

```
10 PRINT"♥":V=53248
20 FORS=832TO895:POKES,255:NEXT
30 FORM=2040TO2042:POKEM,13:NEXT
40 POKEV+21,7:POKEV+39,1:POKEV+40,7
50 POKEV+41,8:POKEV,24:POKEV+1,50
60 POKEV+16,0:POKEV+2,34:POKEV+3,60
70 POKEV+4,44:POKEV+5,70
```

Beyaz yaratığın, sarı ve turuncunun üzerinde olduğunu göreceksiniz. Yaratıkları hareket ettirmede bu öncelik işleyişinden yararlanabilirsiniz.

YARATIKLARI ÇİZELİM

Commodore 64'de bir yaratığı çizmek, bir boyama kitabındaki boşlukları boyamak gibidir. Yaratıklar "piksel" olarak adlandırılan ufacık noktalardan oluşurlar. Bir yaratık çizmek için tüm yapacağınız bu piksellerin bazılarını boyamaktır.

Şekil 3-6. bir boş yaratıcı gösteriyor. Bu şekli yaratık oluşturma tablosu olarak adlandırıyoruz.

1								1								1							
2	6	3	1					2	6	3	1					2	6	3	1				
8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1

Şekil 3-6. Yaratık oluşturma şeması.

Her küçük "kare" yaratık içindeki bir pikseli gösterir. Enine 24 piksel, boyuna 21 piksel vardır ya da bir yaratığın tamamı 504 piksel ile gösterilir. Yaratığın bir şeye benzemesi için. Özel bir program yardımıyla bu pikselleri boyamanız gerekir ... Fakat 500 tek pikselin üzerinde, kontrolü nasıl sağlayabilirsiniz? işte size, bilgisayar programları ile halledebileceğiniz bir konu! 504 sayı girmek yerine, her yaratık için yalnızca 63 sayı girmek zorundasınız. Şimdi bunun nasıl işlediğine bakalım ...

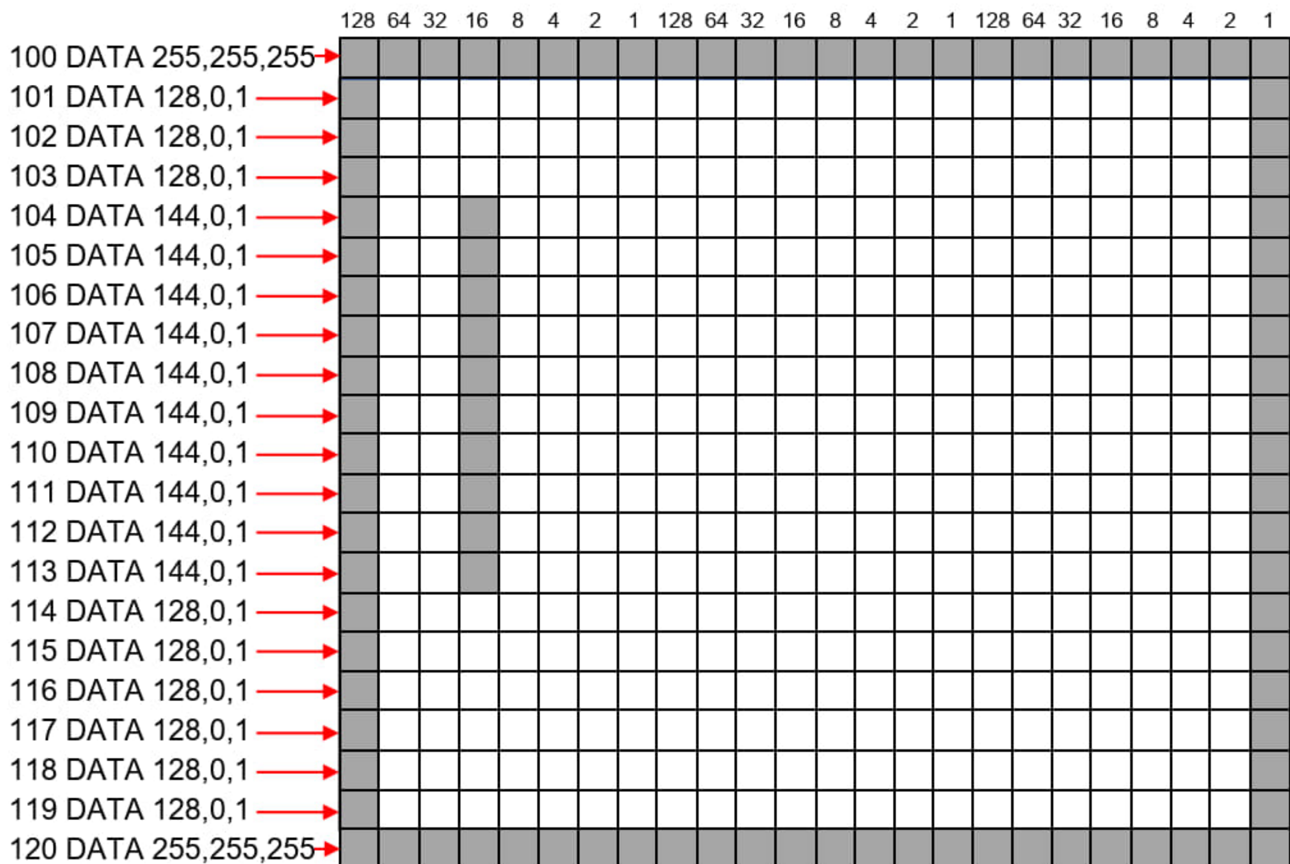
BİR YARATIĞIN AŞAMA AŞAMA OLUŞTURULMASI

Bu; aşamalar halinde verdiğimiz çizim rehberi, kendi yaratıklarınızı oluşturmanızı oldukça kolaylaştıracaktır.

AŞAMA 1:

Burada verdiğimiz yaratık oluşturma programını bir kâğıda yazın. 100 numaralı satırın programınızın özel bir bölümünü oluşturan DATA bölümüyle başladığına dikkat edin. Bu satırda yaratığınızı oluşturacak 63 tane sayı girmelisiniz.

```
10 PRINT"♥":POKE 53280,5:POKE 53281,6
20 V=53248:POKEV+34,3
30 POKE 53269,4:POKE 2042,13
40 FORN=0TO62:READQ:POKE832+N,Q:NEXT
100 DATA 255,255,255
101 DATA 128,0,1
102 DATA 128,0,1
103 DATA 128,0,1
104 DATA 144,0,1
105 DATA 144,0,1
106 DATA 144,0,1
107 DATA 144,0,1
108 DATA 144,0,1
109 DATA 144,0,1
110 DATA 144,0,1
111 DATA 144,0,1
112 DATA 144,0,1
113 DATA 144,0,1
114 DATA 128,0,1
115 DATA 128,0,1
116 DATA 128,0,1
117 DATA 128,0,1
118 DATA 128,0,1
119 DATA 128,0,1
120 DATA 255,255,255
200 X=200:Y=100:POKE53252,X:POKE53253,Y
```



AŞAMA 2:

Yaratık-oluşturma tablosunun piksellerini boyayın (bunun yerine grafik kâğıdı da kullanabilirsiniz. Bir yaratık için 24 yatay, 21 düşey kare gerektiğini unutmayın). Size önerimiz, bir kurşun kalemle hafifçe çizmenizdir, böylece aynı, kâğıdı tekrar kullanmanız mümkün olabilir. İstediğiniz görüntüyü çizebilirsiniz, ancak biz yine örnek olarak basit kutuların çizimini alacağız.

AŞAMA 3:

Şimdi ilk 8 piksele bakın. Piksellerin her sütununda bir sayı olduğunu göreceksiniz (128, 64, 32, 16, 8, 4, 2, 1). Size şimdi ikili aritmetik tipi özel bir toplama göstereceğiz. Birçok bilgisayar bu yöntemi özel bir sayma biçimi olarak kullanır.

128	64	32	16	8	4	2	1

AŞAMA 4:

Dolu olan piksel numaralarını toplayın. 8'lik piksellerden ilk gurubu tamamen dolu, o halde toplam 255'e eşittir.

AŞAMA 5:

Bulduğunuz sayıyı (255) aşağıda verilen yaratık-oluşturma programının 100'üncü satırındaki ilk DATA (veri) yönergesine girin. İkinci ve üçüncü 8'lik gruplar için de, girilmesi gereken sayı 255'tir.

AŞAMA 6:

Yaratığın ikinci sırasının ilk sekiz pikseline bakın. Bu piksellerden sadece birisi doludur. Bu durumda toplamı 28'dir. Bu sayıyı 101 sayılı satıra ilk DATA olarak girin.

128	64	32	16	8	4	2	1

AŞAMA 7:

Diğer 8'lik piksel grubunun değerlerini toplayın ve bu değeri 101'inci satıra girin (bu toplam 0'dır, çünkü geriye kalanların hepsi BOŞTUR). Şimdi diğer gruplara geçin ve aynı işlemleri 8'lik piksel gruplarının her biri için tekrarlayın (21 sıra ve her sırada da 3 grup vardır). Bu işlemlerin sonucunda toplam; 63 sayı tarafından gösterilmiş olur. Daha değişik bir yaklaşımla programa bakarsak; programın her satırının yaratığın bir sırasına karşılık geldiğini çıkarabiliriz. Bir sırada yer alan 3 sayıdan her biri 8'lik piksel gruplarından birini gösterir.

Bu sayılar, bilgisayara hangi piksellerin dolu, hangi piksellerin boş olması gerektiğini bildiren sayılardır.

AŞAMA 8:

Aşağıdaki örnek programda gösterildiği gibi tüm DATA yönergelerini birlikte çalıştırarak (RUN) programınızı daha küçük bir yere sığdırın. Anımsarsanız, size yaratık programınızı bir kâğıda yazmanızı söylemiştik. Bunu söylememizin bir nedeni vardı. Daha önce verilen programda, (100-120) satır numaralı DATA yönergelerinin ayrı ayrı verilmesinin amacı; size yaratığınızdaki hangi piksel grubunun hangi sayıyla ilişkili olduğunu daha açık gösterebilmekti. Programınızın son hali şöyle olmalıdır:

```
10 PRINT"♥":POKE 53280,5:POKE 53281,6
20 V=53248:POKEV+34,3
30 POKE 53269,4:POKE 2042,13
40 FORN=0TO62:READQ:POKE832+N,Q:NEXT
100 DATA 255,255,255,128,0,1,128,0,1,128
,0,1,144,0,1,144,0,1,144,0,1,144,0,1
101 DATA 144,0,1,144,0,1,144,0,1,144,0,1
,144,0,1,144,0,1,128,0,1,128,0,1,128,0,1
102 DATA 128,0,1,128,0,1,128,0,1,255,255
,255
200 X=200:Y=100:POKE53252,X:POKE53253,Y
```

YARATIKLARIN HAREKETLENDİRİLMESİ

Artık yaratıklarınız olduğuna göre bir takım ilginç şeyler yapabiliriz. Yaratıkların ekranda yavaşça hareket etmelerini istiyorsanız şu iki satırı programınıza ekleyiniz:

```
50 POKEV+5,100:FORX=24TO255:POKEV+4,X:NE
XT:POKE V+16,4
55 FORX=0TO65:POKEV+4,X:NEXTX:POKEV+16,0
:GOTO50
```

SATIR 50, Y konumunu 100'e yerleştirir. (50'yi ya da 229'u da deneyebilirsiniz.) Daha sonra ise bir FOR ... NEXT döngüsü oluşturur. Bu döngü yaratığı 0-255 arasındaki X konumuna yerleştirir. 255 'inci konuma gelindiği an ise, ekranın sağ yanına geçebilmesini sağlamak amacıyla sağ X konumunu yerleştirir (POKE V+16,4).

SATIR 55, bir FOR ... NEXT döngüsü oluşturur. Bu döngü de yaratığın ekran üzerindeki son 65 konuma yerleştirilmesini sağlar. Burada X değerleri tekrar 0'a eşitlenir. Bunun nedeni sağ X yerleşimini (POKE V+16,2) kullanmanız, yani; X değerlerinin ekranın en sağından ileride yer almasıdır.

Bu satır, kendisine dönerek (GOTO 50) bir döngü oluşturur. Eğler yaratığınızın, bir kere görünüp sonra hemen yok olmasını istiyorsanız GOTO 50 yönergesini çıkartmanız gerekir. Aşağıda verdiğimiz satırlar ise: yaratığın öne ve arkaya hareket etmesini sağlayacaktır.

```
50 POKEV+5,100:FORX=24TO255:POKEV+4,X:NE
XT:POKEV+16,4:FORX=0TO65:POKEV+4,X:NEXTX
55 FORX=65TO0STEP-1:POKEV+4,X:NEXT:POKEV
+16,0:FORX=255TO24STEP-1:POKEV+4,X:NEXT
60 GOTO50
```

Bu programın nasıl işlediğini anladınız mı? Bu da tıpkı bir önceki gibi. Tek farkı ekranın sağ tarafının sonuna gelindiğinde kendisini ters çevirerek, öbür yönde gitmesi. Bu işleyişi sağlayan ise 1'inci aşamadır. 1'inci aşama programa, yaratığı ekranın sağ tarafındaki 65'ten 0'a kadar olan X değerlerine yerleştirmesini bildirir, daha sonra ise ekranın sol tarafında 255 'ten 0'a doğru, her seferinde -1 ekleyerek 1 konum geriye gidilir.

DÜŞEY KAYMA (VERTICAL SCROLLING)

Yaratıkların bu tip hareketlerine "kayma" (scrolling) denir. Yaratıklarınız, sadece 1 satır kullanarak Y yönünde yukarı veya aşağıya doğru kaydırmanız mümkündür. Aşağıda gösterildiği gibi, 50 ve 55'inci satırlardaki yönergeleri **RETURN** tuşuna basarak silin. Satır numaralarını ise üzerinden geçerek tekrar yazın.

50 (**RETURN**)
55 (**RETURN**)

Şimdi 50 satır numaralı yönergeyi tekrar girin:

```
50 POKE V+4,24:FOR Y=0T0255:POKE V+5,Y:N
EXT
```

BİR YARATIK PROGRAMI ÖRNEĞİ: DANS EDEN FARE

Programcının el kitabında anlatılan teknikler, kimi zaman zor görünebilir. Bunu düşünerek tüm anlatılanları, eğlendirici bir yaratık programında bir araya getirdik. Programın adı Michael'in Dans eden Faresi. Programda 3 sevimli değişik yaratık var. Bu arada canlandırmada ses efektleri yer alıyor. Ayrıca, program yapısının nasıl kurulduğunu ve programın nasıl çalıştığını anlamanıza yardımcı olmak amacıyla her komutun ayrı ayrı açıklamasını da verdik.

```
5 S=54272:POKES+24,15:POKES,220:POKES+1,
68:POKES+5,15:POKES+6,215
10 POKES+7,120:POKES+8,100:POKES+12,15:P
OKES+13,215
15 PRINT"♥":V=53248:POKEV+21,1
20 FORS1=12288TO12350:READQ1:POKES1,Q1:N
EXT
25 FORS2=12352TO12414:READQ2:POKES2,Q2:N
EXT
30 FORS3=12416TO12478:READQ3:POKES3,Q3:N
EXT
35 POKEV+39,15:POKEV+1,68
40 PRINTTAB(160)"■BEN DANS EDEN BIR FARE
YIM!■"
45 P=192
50 FORX=0 TO 347 STEP 3
55 RX=INT(X/256):LX=X-RX*256
60 POKEV,LX:POKEV+16,RX
70 IF P=192 THEN GOSUB 200
```

```
75 IF P=193 THEN GOSUB 300
80 POKE2040,P:FORT=1TO60:NEXT
85 P=P+1:IFP>194THENP=192
90 NEXT
95 END
100 DATA 30,0,120,63,0,252,127,129,254,1
27,129,254,127,189,254,127,255,254
101 DATA 63,255,252,31,187,248,3,187,192
,1,255,128,3,189,192,1,231,128,1,255,0
102 DATA 31,255,0,0,124,0,0,254,0,1,199,
32,3,131,224,7,1,192,1,192,0,3,192,0
103 DATA 30,0,120,63,0,252,127,129,254,1
27,129,254,127,189,254,127,255,254
104 DATA 63,255,252,31,221,248,3,221,192
,1,255,128,3,255,192,1,195,128,1,231,3
105 DATA 31,255,255,0,124,0,0,254,0,1,19
9,0,7,1,128,7,0,204,1,128,124,7,128,56
106 DATA 30,0,120,63,0,252,127,129,254,1
27,129,254,127,189,254,127,255,254
107 DATA 63,255,252,31,221,248,3,221,192
,1,255,134,3,189,204,1,199,152,1,255,48
108 DATA 1,255,224,1,252,0,3,254,0
109 DATA 7,14,0,204,14,0,248,56,0,112,11
2,0,0,60,0,-1
200 POKES+4,129:POKES+4,128:RETURN
300 POKES+11,129:POKES+11,128:RETURN
```

SATIR 5

S = 54272	S değişkenini ses çipinin, bellekteki yerinin başlangıç adresi olan 54272'ye eşitler. Bundan böyle, belleğe S ile birlikte bir değer yerleştireceğiz. (POKES + 24,15 gibi.)
POKES + 24,15	POKE 54296,15 yönergesi gibidir. İşlevi, ses seviyesini en yüksek düzeye çıkarmaktır.
POKES,220	POKE 54272,220 yönergesi ile aynıdır. İşlevi 1'inci sesi 6'ncı oktavdaki kalın sol notasına yerleştirmektir.
POKES + 1,68	POKE 54273,68 yönergesi ile aynıdır. İşlevi: 6'ncı oktavda kalın sol notasına yaklaşan 1'inci sese, yüksek frekans sağlamaktadır.
POKES + 5,15	POKE 54277,15 ile aynıdır ve ses 1 için yükselme/düşmeyi tanımlar ve bu durum "eko" etkisi yaratan yükselme olmadan maksimum düşme düzeyini içerir.
POKES + 6,215	POKE 54278,215 ile aynıdır ve ses 1 için uzatma/bırakma ayarını yapar (215 değeri, uzatma ve bırakma değerlerinin bir kombinasyonunu temsil eder).

SATIR 10

POKES + 7, 120	2'nci sese düşük frekans sağlayan POKE 54279, 120 yönergesi ile aynıdır.

POKES + 8,100	2'nci sese yüksek frekans sağlayan POKE 54280, 100 yönergesi ile aynıdır.
POKES +12,15	POKE 54284,15 komutu ile aynıdır ve 2'nci sesin yükselme/ düşme durumunu, yukarıdaki 1'inci ses ile aynı düzeye getirir.
POKES + 13,215	POKE 54285,215 komutu ile aynıdır ve 2'nci sesin uzatma/ bırakma durumunu, yukarıdaki 1'inci ses ile aynı düzeye getirir

SATIR 15

PRINT"SHIFT CLR/HOME": V=53248	Program başladığı an ekranı temizler. V değişkenini yaratıkları kontrol eden VIC çipinin başlangıç adresi olarak tanımlar.
POKEV + 21,1	Yaratık-1'in görünmesini sağlar.

SATIR 20

FORS1 = 12288 TO 12350	Bu canlandırmada sadece bir yaratık (yaratık-0) kullanacağız. Bununla birlikte üç değişik şekli tanımlamak için: 3 yaratık veri seti kullanmanız gerekiyor. Canlandırma için göstergeçleri yaratık-0 için, bellekte yer alan 3 yerleşime çevireceğiz. Bu yerler 3 değişik şeklimizi tanımlayan verilerin saklandığı yerlerdir. Aynı yaratık 3 farklı yaratık oluşturmak üzere tekrar tekrar tanımlanır. Veri yönergelerinde (data statements) düzinelerce yaratık biçimini tanımlayabilirsiniz. Gördüğünüz gibi, bir yaratığı bir biçimle ya da bir biçimi bir yaratıkla sınırlandırmak zorunda değilsiniz. Göstergeçi farklı biçimler oluşturan verilerin saklandığı belleğin, farklı yerlerine yerleştirmekle bir yaratığa birçok biçim verebilirsiniz. Bu satır Yaratık-1 için verdiğimiz verileri (12288-12350) sayılı adresler arasına yerleştirdiğimiz anlamına gelir.
READ Q1	63 tane sayının 100'üncü satırdan başlayan DATA yönergelerinden sırayla okunmasını sağlar. Değişken adının mutlaka Q1 olması gerekmez. Q1 yerine A ya da Z1 gibi herhangi bir sayısal değişken de kullanılabilirdi.
POKES1,Q1	DATA yönergesinden okuduğu ilk sayıyı (ilk "Q1", 30'dur) ilk bellek adresine (ilk bellek adresi 12288'dir) yerleştirir. Bu, POKE 12288,30 ile aynıdır.
NEXT	Bilgisayara, FOR ve NEXT arasındaki komutları (READQ1 ve POKES1,Q1,NEXT uygulandıktan sonra S1'in artan değerine göre işleme sokulurlar) işleme sokabilmek için döngü değerini kontrol etmesini ve aradaki işlemleri yapmasını söyler. Diğer bir deyişle, NEXT yönergesi, bilgisayarın data yönergelerinde bir sonra gelen (NEXT) Q1 değerini okumasını (READ) sağlar. Bu değer 0'dır. S1'i 1 arttırır ve S1'in değeri 12289'a eşit olur. Sonuçta, POKE 12289,0 yönergesi işlenir. NEXT komutu serideki en son değere ulaşılmasını sağlayıncaya, yani POKE 12350,0 yönergesi işleninceye kadar döngü devam eder.

SATIR 25

FORS2 = 12352 TO 12414	Yaratık-0'ın ikinci biçimi, 12352 ve 12414 adresleri arasına DATA yönergesinden alınarak yerleştirilmiş verilerle tanımlanır. Dikkat ederseniz, 12351 adresinin atlandığını göreceksiniz. Bu ilk yaratık grubunun tanımında kullanılan 64'üncü adrestir ve hiçbir yaratık için veri numarası içermez. Burada hatırlamanız gereken yaratıkları ardışık adreslerde tanımlamak için 64 yerleşim kullanırken, yaratık verilerini sadece ilk 63 adrese yerleştirebilirsiniz.
READ Q2	İlk yaratık biçimini tanımlarken kullandığımız sayıları izleyen 63 sayıyı okur. Bu READ, DATA (veri) alanında hemen sonra gelen sayıyı arar ve her seferinde 63 sayıyı okumaya devam eder.
POKES2,Q2	İkinci yaratık biçiminiz için bellek adreslerine (S2) veri yerleştirir. Adres kullanımı 12352'den başlar.
NEXT	İşlevi 20 satır numaralı yönergeyle aynıdır.

SATIR 30

FORS3 = 12416 TO 12478	Yaratık 0'ın üçüncü biçimi 12416 ve 12478 sayılı adresler arasına yerleştirilmiş veriler tarafından tanımlanır.
READ Q3	Q3 olarak tanımlanan son 63 sayıyı sırayla okur.
POKES2,Q2	Bu sayıları 12416 ve 12478 sayılı adresleri arasına yerleştirir.
NEXT	İşlevi 20 satır numaralı yönergeyle aynıdır.

SATIR 35

POKEV + 39,15	Yaratık 0'ın açık gri renkte olmasını sağlar.
POKEV + 1,68	Yaratık karesinin sağ üst köşesini 68'inci düşey konuma (Y) yerleştirir. 50'nci konum ise karşılaştırmayı kolaylaştırmak açısından ekranın görülebilir alanının sol üst köşesinde yer almıştır.

SATIR 40

PRINT TAB (160)	Ekranın sol üst köşedeki karakter boşluğundan (character space) başlayarak 160 boşluğun atlanmasını sağlar. Bu clear (silme) komutunun altındaki 4 sıra gibidir. PRINT mesajı ekranda 6'ncı satırdan başlar.
"CTRL + WHT"	Eliniz CTRL tuşunun üzerindeyken WHT tuşuna basın. Bu işlemi tırnak işaretlerinin içinde yapmakla negatif bir E (E) elde edeceksiniz. Böylece PRINT edilmekte olan her şey bundan sonra beyaz renkte görünecektir.
BEN DANS EDEN FAREYIM!	Bu bir PRINT yönergesidir. Bu PRINT yönergesi sona erince renkleri yine açık maviye çevirir. C= tuşuna basarken 7 tuşuna basmanız (tırnak işareti kullanarak) negatif bir elmas (◆) şeklinin görünmesini sağlayacaktır.

SATIR 45

P = 192	P değişkenini 192'ye eşitler. 192 kullanmanız gereken göstergeç numarasıdır. Bu, yaratık-0'ı 12288 sayılı adreste başlayan bellek adreslerinde "göstermek" anlamına gelir. Tek yaratık kullanarak 3 ayrı biçim elde etmenin sırrı bu göstergeçi diğer iki yaratık biçiminin bulunduğu yerleşimlere çevirmektir.

SATIR 50

FORX = 0 TO 347 STEP 3	Yaratığınızın bir defada 3X konumu kadar ilerlemesini sağlar. (0'ıncı konumdan 347'nci konuma). Böylece hareket hızlanmış olur.

SATIR 55

RX = INT(X/256)	RX, X/256 işleminin tamsayı halidir. Bunun anlamı şudur: Eğer X, 256'dan küçükse 0'a yuvarlanır. Ancak eğer X, 256 konumuna geldiyse RX'in değeri 1 olacaktır. Biz RX'i ekranın sağ tarafını açmak (ON) istediğimiz an POKE V + 16'ya ekleyeceğimiz 0 ya da 1 sayılarını sağlamak amacıyla kullanacağız.
LX = X-RX*256	Yaratık 0'ıncı X konumundayken formülün görünümü şöyle olacaktır: LX = 0-(0*256) veya 0'ıncı yaratık 1'inci X konumunda iken ise formül şuna dönüşür: LX = 1-(0*256) ya da 1'inci yaratık 256'ncı X konumundaysa: LX = 256-(1*256) veya 0 olmalıdır. Bu X'i tekrar 0 yapar ve ekranın yine sağ tarafından başlamanızı sağlar. (POKEV + 16,1)

SATIR 60

POKEV,LX	V'yi bir değerle birlikte yerleştirerek yaratık-0 için ekranda yatay (X) konumunu yerleştirmiş olursunuz. (Bkz. yaratık oluşturma çizelgesi) Yukarıda gösterildiği gibi yaratığın X konumunu gösteren LX 0'dan 255'e kadar değer alabilir. 255 sayılı konuma geldiği anda ise satır 55'te kurduğumuz LX eşitliği nedeniyle otomatik olarak 0'a eşitlenir.
POKEV + 16,RX	POKEV + 16'nın işlevi her zaman, 256'ıncı konumun ötesinde bulunan ekranın sağ tarafını açmak (ON) ve yatay konumlama koordinatını yeniden değerlemektir. RX: satır 55'te verilen formülle hesaplanan yaratık konumuna bağlı olarak 1 ya da 0 değerini alabilir.

SATIR 70

IFP = 192 THEN GOSUB 200	Yaratık göstergeci 192 ise (ilk yaratık biçimi) ilk ses efekti için dalga-biçimi kontrolü 200 numaralı yönergede yer alan 129 ve 128'e eşitlenir.

SATIR 75

IFP = 193 THEN GOSUB 300	Yaratık göstergeci 193 ise (ikinci yaratık biçimi) ikinci ses efekti için dalga-biçimi kontrolü 300 sayılı yönergede yer alan 129 ve 128'e eşitlenir.

SATIR 80

POKE 2040,P	Yaratık göstergecini 192 sayılı adrese yerleştirir (45 numaralı satırda yer alan P = 192 eşitliğini hatırlayın. P'yi kullandığımız nokta burasıdır).

FORT = 1 TO 60 : NEXT	Basit bir zaman geciktirme döngüsüdür. Hangi fare dans ediyorsa ona hız verir. (60 sayısından fazla ya da az değerler vererek daha hızlı ya da daha yavaş hareketler elde edebilirsiniz.)

SATIR 85

P = P + 1	Şimdi yaptığımız, P değişkeninin orijinal değerine 1 ekleyerek göstergeçin değerini yükseltmektir.
IF P > 194 THEN P = 192	Yaptığımız; yaratığı 3 bellek adresinde göstermektir. 192, 12288-12350 arası adresi; 193, 12352-12414 arası adresi: 194 ise 12416-12478 arası adresi gösterir. Bu satır bilgisayara P 195 olur olmaz P'yi yeniden 192'ye değerlemesini bildirir. Böylece P asla 195 olamaz. P; 192, 193 ve 194 değerlerini aldıktan sonra 192'ye geri döner ve göstergeç DATA içeren 64 baytlık 3 grup içinde yer alan 3 yaratığı ardışık olarak gösterme işlemini tamamlar.

SATIR 90

NEXT X	Yaratık DATA tarafından tanımlanan 3 yaratık biçiminden birini aldıktan sonra onun hareket ettirilmesini sağlar. Fare 1 defada 3 X konumu birden atlayacaktır. (Her defada bir konum atlatarak yavaşça kayması da sağlanabilirdi.) Her defasında farenin 3 adım atması ekranda dans etmesini de getirecektir. NEXT X, 50 numaralı satırda yer alan X konum döngüsü FOR'u kontrol eder.

SATIR 95

END	Fare ekrandan çıktığı an, programı sona erdirir.

SATIR 100-109

DATA	Yaratıklar sırayla DATA sayılarından okunurlar. Önce 1'inci yaratığın biçimini oluşturan 63 sayı, daha sonra ikincisi için ikinci 63 sayı okunur. Bu data, sürekli olarak 3 bellek adresine okunur. Programın bütün yapacağı yaratık-0'ı bu 3 bellek yerleşiminde göstermektir. Bir defada yaratığı üç yerleşimde birden göstermemiz bu canlandırmayı oluşturmamızı sağlıyor. Bu sayıların tüm yaratıklar üzerinde yapacağı değişikliği görmek istiyorsanız 200 sayılı satırın ilk uç sayısını 255 yapın. Daha fazla bilgi için yaratık biçimlerinin tanımlandığı kısma bakmanız yararlı olacaktır.

SATIR 200

POKES + 4,129	Dalga biçimi kontrolüne 129 yerleştirerek ses efektini açar.
POKES + 4,128	Dalga biçimi kontrolüne 128 yerleştirerek ses efektini kapatır.
RETURN	Programı 70'inci satırın sonuna geri yollar.

SATIR 300

POKES + 11,129	Dalga biçimi kontrolüne 129 yerleştirerek ses efektini açar.
POKES + 11,128	Dalga biçimi kontrolüne 128 yerleştirerek ses efektini kapatır.
RETURN	Programı 75'inci satırın sonuna geri yollar.

YARATIK OLUŞTURMA TABLOSU

	YARATIK 0	YARATIK 1	YARATIK 2	YARATIK 3	YARATIK 4	YARATIK 5	YARATIK 6	YARATIK 7
Yaratığı görünür kılma	V+21,1	V+21,2	V+21,4	V+21,4	V+21,16	V+21,32	V+21,64	V+21,128
Belleğe yerleştirme (Göstergeçleri tanımlama)	2040,192	2041,193	2042,194	2043,195	2044,196	2045,197	2046,198	2047,199
Yaratık pikseli için yerleşimler (12288-12798)	12288'den 12350'e	12352'den 12414'e	12416'dan 12478'e	12480'den 12542'e	12544'den 12606'a	12608'den 12670'e	12872'den 12734'e	12736'dan 12798'e
Yaratık rengi	V+39,C	V+40,C	V+41,C	V+42,C	V+43,C	V+44,C	V+45,C	V+46,C
Sol X konumunu tanımlama (0-255)	V+0,X	V+2,X	V+4,X	V+6,X	V+8,X	V+10,X	V+12,X	V+14,X
Sağ X konumunu tanımlama (0-255)	V+16,1 V+0,X	V+16,2 V+2,X	V+16,4 V+4,X	V+16,8 V+6,X	V+16,16 V+8,X	V+16,32 V+10,X	V+16,64 V+12,X	V+16,128 V+14,X
Y konumunu tanımlama	V+1,Y	V+3,Y	V+5,Y	V+7,Y	V+9,Y	V+11,Y	V+13,Y	V+15,Y
Yaratığı yatay yönde genişletme / X	V+29,1	V+29,2	V+29,4	V+29,8	V+29,16	V+29,32	V+29,64	V+29,128
Yaratığı dikey yönde genişletme / Y	V+23,1	V+23,2	V+23,4	V+23,8	V+23,16	V+23,32	V+23,64	V+23,128
Çok-renkli moda geçmek	V+28,1	V+28,2	V+28,4	V+28,8	V+28,16	V+28,32	V+28,64	V+28,128
Çok-renkli 1 (İlk renk)	V+37,C	V+37,C	V+37,C	V+37,C	V+37,C	V+37,C	V+37,C	V+37,C
Çok-renkli 2 (İkinci renk)	V+38,C	V+38,C	V+38,C	V+38,C	V+38,C	V+38,C	V+38,C	V+38,C
Yaratığın önceliklerini belirleme	Kural şudur. Düşük numaralı yaratıkların, yüksek numaralı yaratıklara göre görüntü öncelikleri vardır. Yani yaratık-0 diğerlerinden en önde, yaratık-7 en arkada görüntülenir.							
Çarpışma (Yaratıkla yaratık)	V+30 IF PEEK(V+30) AND X = X THEN [hareket]							
Çarpışma (Yaratıkla zemin)	V+31 IF PEEK(V+31) AND X = X THEN [hareket]							

YARATIK OLUŞTURMA NOTLARI

KASET ARA DEPOSU (BUFFER) İLE BİRLİKTE KULLANILABİLECEK ALTERNATİF YARATIK BELLEK GÖSTERGEÇLERİ VE YERLEŞİMLERİ

BELLEK YERİ (GÖSTERGEÇ)	YARATIK-0 2040,13	YARATIK-1 2041,14	YARATIK-2 2042,15	
Blok 13-15 için yaratık piksel yerleşimleri	832-894	896-958	960-1022	Eğer 1'den 3'e kadar olan yaratıkları kullanıyorsanız, kaset ara deposundaki (832'den 1023'e kadar) bellek yerleşimini kullanabilirsiniz. Fakat 3'ten fazla yaratık için 12288'den 12798'e kadar olan yerleşimleri kullanmanızı öneririz. (Bkz. Tablo)

YARATIKLARIN GÖRÜNTÜLENMESİ

POKE V + 21 ve tablodan bulacağınız bir sayıyı kullanarak herhangi bir yaratığın görüntülenmesini sağlayabilirsiniz. Ancak bir yaratığı görüntülemeniz diğerlerinin kaybolmasına neden olacaktır. İki ya da daha çok yaratığı aynı anda görüntülemek için, istediğiniz yaratık numaralarını toplayarak tek sayı halinde kullanmalısınız. (Örnek: POKE V + 21, 6 yönergesi hem yaratık-1'in, hem de yaratık-2'nin görüntülenmesini sağlayacaktır.

Şimdi size, diğer yaratıkların kaybolmasına neden olmadan, bir yaratığın kaybolmasını sağlayacak yöntemi veriyoruz. (Bu yöntem, canlandırmalarınız için yararlı olacaktır.)

Örnek: Sadece Yaratık-0'ın kaybolmasını istiyorsanız şu yönergeyi girin: POKE V + 21, PEEK V + 21 AND (255-1)

(255-1)'de yer alan 1 sayısını 1, 2, 4, 8, 16, 32, 64 ya da 128'le değiştirin (0'dan 7'ye kadar olan tüm yaratıklar için). Bir yaratığı tekrar görüntülemek ve bu arada diğerlerini etkilememek istiyorsanız POKEV + 21, PEEK (V + 21) OR 1 yazın. Yaratığa bağlı olarak OR1'i, OR2 ile (yaratık-2) ya da OR4 (yaratık-3) ile değiştirebilirsiniz. Bu işleyiş tüm yaratıklar (0-7) için geçerlidir.

255'İN İLERİSİNDE BULUNAN X KONUMU DEĞERLERİ

X konumu, 0'dan başlar, 255'te biter. 255'in ilerisinde yer alan diğer X konumları için yine 0'dan başlar, 255'te biter. Bir yaratığı ekranın sağ tarafında 255'ten ilerideki X konumlarına yerleştirmek için, önce POKEV + 16 yazacaksınız. Daha sonra ise 0'la 63 arasında yeni bir X değeri yerleştireceksiniz (POKE ile). Böylece yaratığınız ekranın sağ yanında bulunan X konumlarından birisine yerleştirilmiş olacaktır. 0-255 konumlarına geri dönmek istiyorsanız, POKEV + 16, 0 yazmalı ve POKE komutu ile 0'la 255 arası bir X değeri kullanmalısınız.

Y KONUMU DEĞERLERİ

Y konumu değerleri 0'dan başlar ve 255'te biter. 0-49 arası değerler ekranın üst kısmında, görülebilir alan dışında, 50 ve 233 arası değerler ekranın görülebilir kısmında. 203-255 arası değerler ekranın alt kısmında, görülebilir alanın dışında kalırlar.

YARA TIK RENKLERİ

Yaratığın beyaz olmasını istiyorsanız POKE V + 39, 1 yazmalısınız (Tabloda yer alan POKE komutunda kullanılacak renkler ve aşağıda gösterilen renk kodlarını kullanın)

0-SİYAH	4-MOR	8-TURUNCU	12-YEŞİLİMSİ
1-BEYAZ	5-YEŞİL	9-KAHVERENGİ	13-AÇIK YEŞİL
2-KIRMIZI	6-MAVİ	10-AÇIK KIRMIZI	14-AÇIK MAVİ
3-TURKUVAZ	7-SARI	11-KOYU GRİ	15-AÇIK GRİ

BELLEK YERLEŞİMİ

Her bir yaratığı tanımlamak için 63 bayt gerekli olduğundan, bilgisayarınızın belleğinde ayrı ayrı 64 baytlık sayı blokları ayırmalısınız. Bellekte bu tür yerleşimleri düzenleyebilmek için, yaratık göstergeçlerini yukarıdaki tabloda verilen değerlere uygun olarak tanımlamalısınız. Böylece her yaratık tek olarak, istediğiniz gibi tanımlanacaktır. Tüm yaratıkların aynı olması için, aynı olmasını istediğiniz yaratıklar için aynı kaydı kullanın.

DEĞİŞİK YARATIK GÖSTERGEÇLERİNİN TANIMLANMASI

Burada vereceklerimiz, sadece konuyla ilgili birtakım önerilerdir.

UYARI: Yaratık göstergeçlerinizi RAM'ın istediğiniz yerine yerleştirebilirsiniz. Ancak eğer bellekte çok altlara yerleştirirseniz, uzun bir BASIC programı yaratıklara ilişkin verilerinizi yok edebilir (bunun tam tersi yani, yaratık verilerinin bir BASIC programının üzerine gelmesi de olasıdır). Yaratık verilerinin BASIC programı yüzünden kaybolmasını önlemek için, yaratık verilerinizi belleğin üst bölümlerine yerleştirebilirsiniz. Örneğin: Yaratık-0 için 2040, 192 sayılarını kullanarak yaratığın 12288-12350 adresleri arasına, yaratık-1 için ise 2041, 193 sayılarını kullanarak 12352-12414 adresleri arasına yerleştirilmelerini sağlayabilirsiniz. Yaratıkların, renk ve biçim bilgilerini aldıkları adresleri ayarlamakla 64 farklı yaratık tanımlamanız ve oldukça uzun bir BASIC programı yazmanız mümkün olabilir. Bunu yapmak için

DATA yönergelerinizde çeşitli yaratık biçimleri tanımladıktan sonra, göstergeçi değiştirerek belirli bir yaratığı yeniden tanımlayın. Böylece yaratığınız farklı yaratık resimleme verileriyle belleğin farklı alanlarına yerleştirilmiş olur. Bu işleyişi dans eden fare programında görebilirsiniz. Eğer iki ya da daha çok yaratığın aynı biçimi almasını istiyorsanız çakıştırmak istediğiniz yaratıklar için aynı göstergeci ve bellek yerleşimini kullanın (hala her yaratığın rengini ve konumunu değiştirmek olanağınız vardır). Örneğin yaratık-0 ve yaratık-1'i, POKE 2040, 192 ve POKE 2041, 192 yönergelerini kullanarak aynı yerleşimi kullanabilirsiniz.

ÖNCELİK

Öncelik, görüntü ekranında bir yaratığın hareket etmek üzere diğerinin önünde ya da üstünde görünmesidir. Burada tek kural var; daha düşük yaratık numarasına sahip olanlar, yüksek numaralı yaratıklara göre öncelik taşırlar. Yaratık-0 diğer tüm yaratıklara göre önceliklidir. Yaratık-7 ise hiçbir yaratığa gere öncelik taşımaz. Yaratık-1, 2-7 numaralı yaratıklara göre önceliklidir vs. İki yaratığı aynı konuma yerleştirirseniz, önceliği fazla olan yaratık her zaman diğerinin önünde görünecektir. Önceliği düşük olan yaratıklar diğerleri tarafından ya örtülür, ya da arkalarında görünürler.

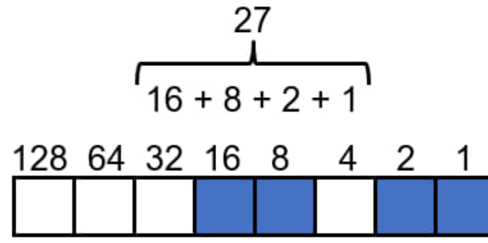
ÇOK-RENKLİ KULLANIM

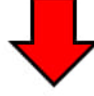
Yaratık resimlerinizde tek bir piksel değil de çift piksel kullanabilirsiniz. Çalışmanız gereken mod: çok renkli moddur (başka bir deyişle: yaratığı oluşturan her renkli nokta ya da blok yan yana iki pikselden oluşmalıdır). Renk konusunda 4 seçeneğiniz var:

Yaratık Rengi (yukarıdaki verdiğimiz tablodan), çok-renkli 1, çok-renkli 2 ve zemin rengi. Bir yaratık resminde 8 piksellik bir yatay blok olduğunu düşünelim. Piksel çiftlerinin rengi, hangi piksel (ya da piksellerin) dolu olduğuna bağlıdır:

☐☐	**ZEMİN**	(Her iki pikselin de boş (0) olması ekranın iç rengini (zemin) verecektir.)
☐■	**ÇOK-RENKLİ MOD 1**	(Piksel çiftinde sağda yer alan pikselin dolu olması her iki pikselin de çok-renkli mod 1'de olmasını sağlayacaktır.)
■☐	**YARATIK RENGİ**	(Piksel çiftinde solda yer alan pikselin dolu olması her iki pikselin de yaratık renginde olmasını sağlayacaktır.)
■■	**ÇOK-RENKLİ MOD 2**	(Her iki pikselin de dolu olması piksel çiftinin çok-renkli 2'de olmasını sağlayacaktır.)

Şimdi aşağıda verdiğimiz 8 piksellik yatay sıraya bakın. Bu blok iki pikselin zemin renginde, ikinci piksel çiftinin çok-renkli 1'de, üçüncü piksel çiftinin yaratık rengi modunda, dördüncü piksel çiftinin ise çok-renkli mod 2'de yer almasını sağlar. Yukarıda gösterildiği gibi piksel çiftlerinin rengi bir çiftte bulunan bitlerden hangisinin dolu, hangisinin boş olduğuna bağlı olarak belirlenir. Piksel çiftlerinin rengine karar verdikten sonra yapacağınız, 8 piksellik blokta yer alan dolu piksellerin değerlerini toplamak ve bu toplamı bellekte uygun bir adrese yerleştirmektir. Örneğin; aşağıda gösterdiğimiz 8 piksellik sıranın 832 sayılı yerleşimden başlayan bir yaratığın ilk bloğu olduğunu düşünelim. Dolu piksellerin değeri; 27'ye eşittir (16 + 8 + 2 + 1 = 27). Yazacağınız POKE yönergesi şöyle olmalıdır: POKE 832, 27.





YARATIKTA GÖRÜNÜMLERİ ŞÖYLE OLACAKTIR



ZEMİN RENGİ	**ÇOK RENKLİ 1**	**YARATIK RENGİ**	**ÇOK RENKLİ 2**

ÇARPIŞMA

Bir yaratığın diğeriyle çarpışıp çarpışmadığını, şu yönergeyi kullanarak anlayabilirsiniz:

```
IF PEEK (V + 30) AND X = X THEN PRINT"YA
RATIK YARATIK ILE CARPISTI"
```

Bu satır belirli bir yaratığın diğer bir yaratıkla çarpışıp çarpışmadığını kontrol etmenizi sağlayacaktır. X konumları şöyle belirlenmiştir: Yaratık-0 için X = 1, yaratık-1 için X = 2, yaratık-2 için X = 4, yaratık-3 için X = 8, yaratık-4 için X = 16, yaratık-5 için X = 32, yaratık-6 için X = 64, yaratık-7 için X = 128.

Bir yaratığın bir zemin karakteriyle çarpışıp çarpışmadığını ise şu satırla kontrol edebilirsiniz:

```
IF PEEK (V + 31) AND 4 = 4 THEN PRINT"ZE
MIN KARAKTERI ILE CARPISMA OLDU"
```

DATA YÖNERGELERİNDE GRAFİK KARAKTERLERİN KULLANIMI

Şimdi vereceğimiz programla, DATA yönergenizde boşlukları ve dolu daireleri kullanarak bir yaratık elde edebilirsiniz. Oluşturulan yaratık ve yaratık veri kayıtlarına POKE komutu ile yerleştirilen sayılar belirtilmiştir.

```
10 PRINT"♥":FOR I=0TO63:POKE832+I,0:NEXT
20 GOSUB60000
999 END
60000 DATA"          ●●●●●●●          "
60001 DATA"        ●●●●●●●●●●●        "
60002 DATA"       ●●●●●●●●●●●●●       "
60003 DATA"       ●●●●●   ●●●●●       "
60004 DATA"      ●●●●● ●●● ●●●●●      "
60005 DATA"      ●●●●● ●●● ●●●●●      "
60006 DATA"      ●●●●● ●●● ●●●●●      "
60007 DATA"       ●●●●●   ●●●●●       "
60008 DATA"       ●●●●●●●●●●●●●       "
60009 DATA"       ●●●●●●●●●●●●●       "
60010 DATA"       ● ●●●●●●●●● ●       "
60011 DATA"        ● ●●●●●●● ●        "
60012 DATA"        ●  ●●●●●  ●        "
60013 DATA"         ●  ●●●  ●         "
60014 DATA"         ●  ●●●  ●         "
60015 DATA"          ●  ●  ●          "
60016 DATA"          ●  ●  ●          "
60017 DATA"           ●●●●●           "
60018 DATA"           ●●●●●           "
60019 DATA"           ●●●●●           "
60020 DATA"           ●●●●●           "
60100 V=53248:POKEV,200:POKEV+1,100
60101 POKEV+21,1:POKEV+39,14:POKE2040,13
60105 POKEV+23,1:POKEV+29,1
60110 FOR I=0TO20:READ A$:FOR K=0TO2:T=0
60111 FOR J=0TO7:B=0
60140 IF MID$(A$,J+K*8+1,1)="Q" THEN B=1
60150 T=T+B*2↑(7-J):NEXT:PRINT T;
60151 POKE 832+I*3+K,T:NEXT:PRINT:NEXT
60200 RETURN
```

BÖLÜM 4

SES VE MÜZİK PROGRAMLAMA

- Giriş
- Volüm Kontrolü
- Ses Dalgalarının Frekansları
- Birden Fazla Sesin Kullanımı
- Dalga Biçimi Değişimleri
- Envelope Üreticisi
- Filtreleme
- İleri Teknikler
- Senkronizasyon Ve Ring Modülasyonu

GİRİŞ

Commodore bilgisayarınız, diğer bilgisayarlarda da bulunan elektronik müzik synthesizer'larının en gelişmişlerinden biriyle donatılmıştır. Commodore 64'ünüz ün ses oluşturma düzeni; tamamen adreslenebilen 3 ses, yükselme/düşme/durma/kaybolma (ADSR), filtreleme, modülasyon ve beyaz gürültüden oluşmuştur. Tüm bu yetileri birkaç basit BASIC ve/veya assembler komutu ve fonksiyonu ile kullanabilirsiniz. Böylece tasarımı görece kolay programlar kullanarak oldukça karmaşık sesler ve şarkılar üretmeniz mümkündür.

Kullanım kılavuzunun bir bölümü Commodore bilgisayarınız içindeki ses ve müzik synthesizer'ı olan 6581 "SID" çipinin yetilerini keşfetmenize yardımcı olacaktır. Size hem müzikal bilginin ardındaki kuramı, hem de bu bilgileri Commodore 64'te gerçek şarkılara dönüştürmenize yardımcı olacak pratik yanları açıklayacağız.

Müzik synthesizer'ınızdan iyi sonuçlar elde etmek için ne deneyimli bir programcı ne de bir müzik üstadı olmanız gerekmez. Bu bölümde bulunan pek çok programlama örneği ve açıklamalar işe başlamanızı kolaylaştıracaktır.

Belleğin belirli yerleşimlerini kullanarak (POKE yönergeleriyle) ses üreticisine ulaşabilirsiniz. Hanelerin tam listesi EK-O'da verilmiştir. Burada her kavramı tüm ayrıntılarıyla açıklayacağız. Bu bölümün sonunda siz de, hemen hemen sonsuz çeşitte denemelerinize başlamaya hazır olacaksınız.

Her bölümün başında bir örnek program ve bunun hemen ardından bu programın satır satır açıklaması verilerek, tartışılan özelliğin nasıl kullanılacağı gösterilecektir. Burada verilen teknik açıklamalar gerçekte nelerin olup bittiğini merak ettiğinizde okumanız içindir.

Ses programınızın motoru POKE komutudur. POKE komutu BEL bellek yerleşimine, belirlenmiş bir NUM değerini yükler:

POKE BEL, NUM

Müzik sentezi için kullanılan bellek yerleşimleri (BEL), Commodore 64'te 54272 ($D400)'den başlar. 6581 (SID) çipi kayıt haritasını kullanırken hatırlamanız gereken, 54272-54296 arası bellek hanelerinin POKE haneleri olduğudur. Bu haneleri kullanmanın bir yolu da sadece 54272 hanesini akılda tutarak buna 0-24 arası bir sayı eklemektir. Böylece SID çipinde size gereken 54272-54296 arası tüm haneleri POKE edebileceksiniz. POKE komutunda kullanacağınız sayı (NUM) ise 0 ile 255 arasında olmalıdır.

Müzik yapma konusunda deneyiminiz biraz artınca PEEK fonksiyonunu kullanarak bu konudaki yetinizi biraz daha derinleştirmiş olacaksınız. PEEK, bellek hanesinde o anda bulunan değeri okumanızı sağlayan bir fonksiyondur.

X = PEEK(BEL)

X değişkeni BEL'inci bellek yerleşiminin o andaki değerine eşitlenir. Tabii ki programınız birçok değişik BASIC komutu da içerecektir. Bunlar için bu kitabın BASIC yönergeleri bölümüne bakmalısınız.

Şimdi sizinle bilgisayarınızda var olan 3 ses olanağından yalnızca birini kullanarak, basit bir program denemesine girişelim.

Bilgisayarınız hazır mı? önce NEW yazarak **RETURN** tuşuna basın aşağıdaki programı girin. Daha sonra, bu programı Commodore datasette, ya da diskte sakladıktan sonra çalıştırın (RUN yazın **RETURN** tuşuna basın).

ÖRNEK PROGRAM 1:

```
5 S=54272
10 FORL=STOS+24:POKEL,0:NEXT
20 POKES+5,9:POKES+6, 0
30 POKES+24,15
40 READHF,LF,DR
50 IFHF<0THENEND
60 POKES+1,HF:POKES,LF
70 POKES+4,33
80 FORT=1TODR:NEXT
90 POKES+4,32:FORT=1TO50:NEXT
100 GOTO40
110 DATA 25,177,250,28,214,250
120 DATA 25,177,250,25,177,250
130 DATA 25,177,125,28,214,125
140 DATA 32,94,750,25,177,250
150 DATA 28,214,250,19,63,250
160 DATA 19,63,250,19,63,250
170 DATA 21,154,63,24,63,63
180 DATA 25,177,250,24,63,125
190 DATA 19,63,250,-1,-1,-1
```

Şimdi bu programı satır satır açıklayalım. Programın anlamadığınızı hissettiğiniz bir bölümü olursa bakacağınız yer burası:

ÖRNEK PROGRAM 1'İN SATIR SATIR AÇIKLAMASI:

SATIRLAR	AÇIKLAMA
5	Ses çipinin başlangıcı için, S tanımlanır.
10	Tüm ses çipi kayıtlarını temizler.
20	Ses 1 için yükselme/düşme (A = 0, D = 9) ve durma/kaybolma (S = 0, R = 0) tanımlanır.
30	Ses düzeyini en yüksek değerine eşitler.
40	Yüksek frekans, alçak frekans ve nota sürelerini (duration) okur.
50	Yüksek frekans 0'dan küçükse şarkıyı bitirir.
60	Ses 1'in yüksek ve alçak frekanslarını tanımlar. (POKE ile)
70	Ses 1 için dişli dalga biçimine geçer.
80	Nota uzunluğu için zaman döngüsü.
90	Ses 1 için dişli dalga biçimini terk eder.
100	Bir sonraki notaya dönüş.
110-180	Şarkı için veri; yüksek frekans, alçak frekans, nota uzunlukları (number of counts).
190	Şarkının son notası, negatif 1 şarkının bittiğini bildirir.

VOLÜM (SES DÜZEYİ) KONTROLÜ:

24 nolu çip kaydı tüm volüm (ses düzeyi) kontrolünü üstlenmiştir. Volüm, 0'dan 15'e kadar ayarlanabilir. Diğer 4 bitin kullanımını ise daha sonra açıklayacağız. Şimdilik volümün, 0'la 15 arasında olduğunu bilmeniz yeterli. Volümü nasıl vereceğinizi, örnek program 1'in 30'uncu satırına bakarak görebilirsiniz.

SES DALGALARININ FREKANSLARI:

Sesi oluşturan, havanın dalgalar biçimindeki hareketidir. Havuza bir taş attığınız zaman dışarıya doğru yayılan dalgaları düşünün. İşte benzeri dalgaları havada oluşturmanız, onları duyabilmenizi de sağlayacaktır. Ardışık iki dalganın en yüksek noktaları arasında geçen zaman, dalganın bir çevrimi (devri) için gerekli zamanı (saniye) verecektir (n = saniye sayısı). Bu sayının tersi (1 /n) ise 1 saniyedeki devir sayısını verir. 1 saniyedeki devir sayısına ise frekans denir. Ses tonunun yüksekliği ya da alçaklığı ise, oluşturulan ses dalgalarının frekansı tarafından belirlenir.

Commodore bilgisayarınız frekansı belirlemek üzere iki hane ayırır. Müzik notalarının sekiz oktavını üretmek için gereksineceğiniz frekans değerlerini EK-E'de bulabilirsiniz. Nota tablosunda yer alanların dışında bir frekans yaratmak için "Fout" (frekans çıktısı)'u kullanmanız gerekir. Aşağıdaki formül, oluşturmak istediğiniz sesin frekansını (F_n) hesaplamanızı sağlayacaktır. Her notanın bir yüksek ve bir alçak frekans numarasına sahip olması gerektiğini unutmayın.

$F_n = F_{out} / .06097$

"Yeni" noktanız için F_n'i belirledikten sonra yapacağınız iş, notanız için bir yüksek ve bir alçak frekans değeri belirlemektir. Bunu yapmak için önce F_n'i yuvarlayıp tam sayı bölümünü almak gerekir. Şimdi F_{hi} = INT (F_n/256) formülünü kullanarak yüksek frekans hanesinin, $F_{lo} = F_n - (256 * F_{hi})$ formülünü kullanarak alçak frekans hanesinin değerlerini bulabilirsiniz.

Bu noktaya kadar, artık bilgisayarınızda bir ses oluşturmayı öğrendiniz. Eğer isterseniz evinizin konser salonunda, bilgisayar orkestranızın "maystro"su olabilir ve sevdiğiniz bir parçanın çalınmasını yönetebilirsiniz.

BİRDEN FAZLA SESİN KULLANIMI:

Commodore bilgisayarınızda birbirinden tamamen bağımsız olarak kontrol edilebilen 3 ses (osilatör) bulunur. İlk örnek programımızda, bunlardan sadece birini kullandık. Daha sonra, bu sesler tarafından oluşturulan sesin niteliklerinin, nasıl değiştirileceğini öğreneceksiniz. Biz şimdilik üç sesi de kullanmaya çalışalım.

Bu örnek, yazılmış bir şarkıyı, bilgisayarınızın orkestrası için düzenleme yollarından birini gösteriyor. Programı olduğu gibi yazın ve kasete ya da diskete saklayın. Programı girmeye başlamadan önce NEW yazmayı unutmayın.

ÖRNEK PROGRAM 2:

```
10 S=54272:FORL=STOS+24: POKEL,10:NEXT
20 DIMH(2,200),L(2,200),C(2,200)
30 DIMFQ(11)
40 V(0)=17:V(1)=65:V(2)=33
50 POKES+10,8:POKES+22,128:POKES+23,244
60 FORI=0TO11:READFQ(I):NEXT
100 FORK=0TO2
110 I=0
120 READNM
130 IFNM=0THEN250
140 WA=V(K):WB=WA-1:IFNM<0THENNM=-NM:WA=
0:WB=0
```

```
150 DR%=NM/128:OC%=(NM-128*DR%)/16
160 NT=NM-128*DR%-16*OC%
170 FR=FQ(NT)
180 IFOC%=7THEN200
190 FORJ=6TOOC%STEP-1:FR=FR/2:NEXT
200 HF%=FR/256:LF%=FR-256*HF%
210 IFDR%=1THENH(K,I)=HF%:L(K,I)=LF%:C(K
,I)=WA:I=I+1:GOTO120
220 FORJ=1TODR%-1:H(K,I)=HF%:L(K,I)=LF%:
C(K,I)=WA:I=I+1:NEXT
230 H(K,I)=HF%:L(K,I)=LF%:C(K,I)=WB
240 I=I+1:GOTO120
250 IFI>IMTHENIM=I
260 NEXT
500 POKES+5,0:POKES+6,240
510 POKES+12,85:POKES+13,133
520 POKES+19,10:POKES+20,197
530 POKES+24,31
540 FORI=0TOIM
550 POKES,L(0,I):POKES+7,L(1,I):POKES+14
,L(2,I)
560 POKES+1,H(0,I):POKES+8,H(1,I):POKES+
15,H(2,I)
570 POKES+4,C(0,I):POKES+11,C(1,I):POKES
+18,C(2,I)
580 FORT=1TO80:NEXT:NEXT
590 FORT=1TO200:NEXT:POKES+24,0
600 DATA 34334,36376,38539,40830
610 DATA 43258,45830,48556,51443
620 DATA 54502,57743,61176,64814
1000 DATA 594,594,594,596,596,1618,587,5
92,587,585,331,336
1010 DATA 1097,583,585,585,585,587,587,1
609,585,331,337,594,594,593
1020 DATA 1618,594,596,594,592,587,1616,
587,585,331,336,841,327
1999 DATA 1607,0
2000 DATA 583,585,583,583,327,329,1611,5
83,585,578,578,578
2010 DATA 196,198,583,326,578,326,327,32
9,327,329,326,578,583
2020 DATA 1606,582,322,324,582,587,329,3
27,1606,583,327,329,587,331,329
2999 DATA 329,328,1609,578,834,324,322,3
27,585,1602,0
3000 DATA 567,566,567,304,306,308,310,15
91,567,311,310,567
3010 DATA 306,304,299,308,304,171,176,30
6,291,551,306,308
3020 DATA 310,308,310,306,295,297,299,30
4,1586,562,567,310,315,311
3030 DATA 308,313,297,1586,567,560,311,3
09,308,309,306,308
3999 DATA 1577,299,295,306,310,311,304,5
62,546,1575,0
```

Aşağıda Örnek Program 2'nin satır satır açıklamasını bulabilirsiniz. Şu anda üç sesin nasıl kontrol edildiğini öğrendik.

ÖRNEK PROGRAM 2'NİN AÇIKLAMASI

SATIRLAR	AÇIKLAMA
10	S değişkenine ses çipinin başlangıç adresini yerleştirir ve tüm kayıtları temizler.
20	Şarkının işlekliğini içerecek dizi boyutlarını tanımlar. Her hane için 1/16'lık ölçü bulunur.
30	Her nota için temel frekans dizilerinin boyutlarını tanımlar.
40	Her ses için dalga boyu kontrol baytlarını içerir.
50	Ses 2 için yüksek darbeyi tanımlar. Filtreleme ve 3'üncü filtre sesi için rezonansı tanımlar.
60	Her nota için temel frekansları okur.
100	Her ses için kod çözme döngüsünü başlatır.
110	Göstericiyi işleklik dizisine ayarlar.
120	Kodlanmış notaları okur.
130	Eğer kodlanmış nota 0 ise bir sonraki sese geçirir.
140	Dalga biçimi kontrolünü uygun sese yerleştirir. Eğer sessizlik gerekiyorsa dalga biçimi kontrolü 0'a eşitlenir.
150	Süre ve oktav kodlarını çözer.
160	Nota kodu çözer.
170	Bu nota için temel frekansı alır.
180	Eğer oktav en yüksek noktadaysa, bölme döngüsüne atlar.
190	Ana (baz) temel frekansı 2 uygun zamana böler.
200	Yüksek ve alçak frekans baytlarını alır.
210	Eğer 16'ncı notadaysa; işleklik dizisi yerleştirir: yüksek frekans, alçak frekans ve dalga biçimi kontrolü (ses açık).
220	Notanın son vuruşu dışındaki vuruşlar için işleklik dizisini yerleştirir; yüksek frekans, alçak frekans ve dalga biçimi kontrolü (ses açık).
230	Notanın son vuruşu için işleklik dizisini yerleştirir; yüksek frekans, alçak frekans ve dalga biçimi kontrolü (ses kapalı).
240	İşleklik dizisi için artımlı gösterici. Bir sonraki notayı alır.
250	Öncekinden daha uzunsa, işleklik sayılarını yeniden değerler.
260	Bir sonraki ses için geri döner.
500	Ses 1 için yükselme/düşme kontrollerini kurar. (A = 0, D = 0) Ses 1 için durma/kaybolma kontrollerini kurar. (S = 15, R = 0)
510	Ses 2 için (yükselme/düşme/durma/kaybolma) kontrollerini yerleştirir. (A = 5, D=5, S = 8, R = 5)
520	Ses 3 için (yükselme/düşme/durma/kaybolma) kontrollerini yerleştirir. (A = 0, D = 10, S = 12, R = 5)
530	Ses düzeyini 15'e eşitler. Low-pass filtering.
540	Her 1/16'lık ölçü için döngüyü başlatır.
550	Tüm sesler için işleklik dizisinden düşük frekans POKE'lar.
560	Tüm sesler için işleklik dizisinden yüksek frekans POKE'lar.
570	Tüm sesler için işleklik dizisinden dalga-biçimi kontrolü POKE'lar.

SATIRLAR	AÇIKLAMA
580	1/16'lık ölçü için zamanlama döngüsü ve bir sonraki 1/16'lık ölçü için geri döner.
590	Ara (pause) sesi kapatır.
600-620	Ana (base) frekans verileri.
1000-1999	Ses 1 verileri.
2000-2999	Ses 2 verileri.
3000-3999	Ses 3 verileri.

Veri yönergelerinde kullanılan değerler, EK-E'de verilen nota tablosundan ve aşağıdaki çizelgeden çıkarılmıştır.

NOTA TİPİ	SÜRE (DURATION)
1/16	128
1/8	256
Noktalı 1/8	384
1/4	512
1/4 + 1/16	640
Noktalı 1/4	768
1/2	1024
1/2 + 1/16	1152
1/2 + 1/8	1280
Noktalı 1/2	1536
Tam	2048

Nota tablosundan bulunacak sayılar yukarıda verilen sürelere eklenir. Böylece her nota, programınız tarafından çözülecek 1 sayı kullanılarak girilir. Bu nota kodlama yöntemlerinden sadece birisidir. Siz daha rahatlıkla kullanabileceğiniz başka yöntemler de geliştirebilirsiniz. Burada bir notayı kodlamak için kullanılan formül şöyledir:

1. Süre (1/16'lık ölçümlerin sayısı) 8 ile çarpılır.
2. 1'inci basamaktan elde edilecek sayı seçilen oktava (0-7) eklenir.
3. 2'nci basamağın sonucu 16 ile çarpılır.
4. 3'üncü basamağın sonucu ile seçilen nota (0-11) toplanır.

Başka bir deyişle:

((((D*8) + O) * 16) + N)

D: Süre (Duration) O: Oktav N: Nota

Sessizlik ise, sürenin tersi kullanılarak elde edilir. (Bir ölçümdeki 1 /16'lıkların sayısı * 128)

ÇOK SESLİLİĞİN KONTROLÜ (MULTIPLE VOICE)

Birden fazla sesi bir arada kullanırken 3 sesin zamanlamasının mutlaka koordine edilmesi gerektiğini anlayacaksınız. Bu programda bu koordinasyonu sağlamak için:

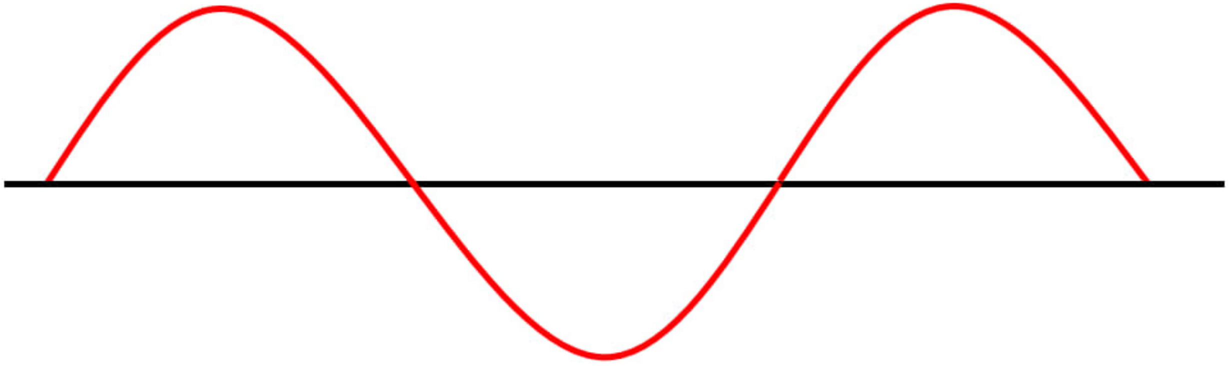
1. Tüm müzikal ölçüler 16 parçaya bölünür.
2. Her 1/16'lık ölçü aralığında yer alan olaylar 3 ayrı dizide depolanır.

Yüksek ve alçak frekans baytların en yüksek oktavın ikiye bölümünden elde edilir (satır 180 ve 190). Dalga biçimi kontrol baytı bir notanın başlaması ya da zaten çalan bir notanın devamı için bir sinyal teşkil eder. Aynı zamanda bir notanın sonu için sinyal olabilir. Her ses için dalga biçimi seçimi 40 sayılı satırda yapılmıştır.

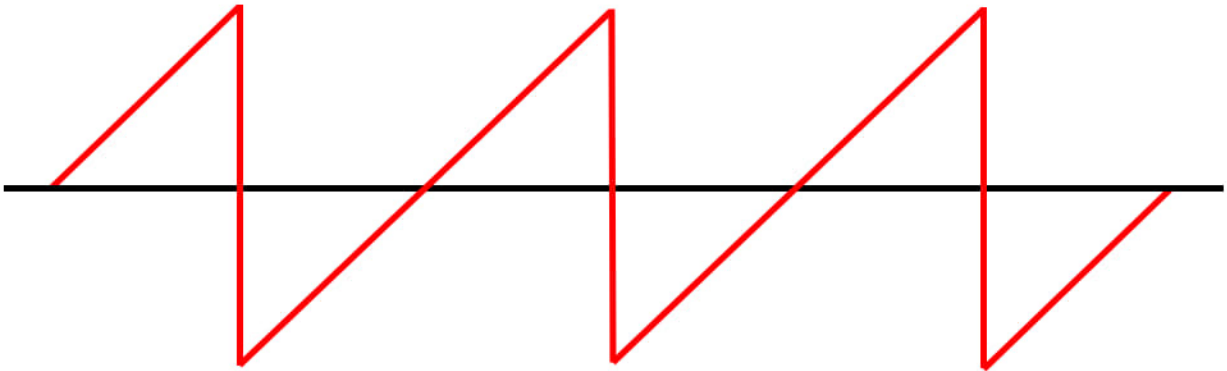
Tekrar edelim: Bu yöntem birden fazla sesin kontrolü için kullanılacak yöntemlerden sadece birisidir. Siz kendi yöntemlerinizi kullanabilirsiniz. Ancak şimdi herhangi bir yazılı notayı ele alıp, her üç ses için gereken notaları şekillendirebiliyor olmalısınız.

DALGA BİÇİMİ DEĞİŞİMLERİ:

Bir sesin tonal niteliğini "TIMBRE" olarak adlandırıyoruz. Bir sesin timbre'ini en temelde belirleyen şey o sesin dalga biçimidir. Suya atılan çakıl taşı örneğini anımsayacaksınız, dalgalar nerdeyse tüm havuza yavaşça yayılırlar. Bu dalgalar şimdi üzerinde konuşacağımız sinüs dalgalarına benzerler. Bunlara kısaca "sin" dalgaları diyeceğiz.



Tartıştıklarımızı biraz daha somut ve pratik hale getirelim. Bunun için farklı dalga biçimlerini incelemek üzere ilk program örneğine dönüyoruz. Bunu yapmamızın nedeni değişimlerin bir seste daha kolay duyulabilmesi. Daha önce yazmış olduğunuz müzik programını teypten ya da diskinizden yükleyin. (LOAD) ve tekrar çalıştırın (RUN). Bu program 6581 SID çipinin ses üretici cihazından alınan (dişli) dalga biçimini kullanıyor.



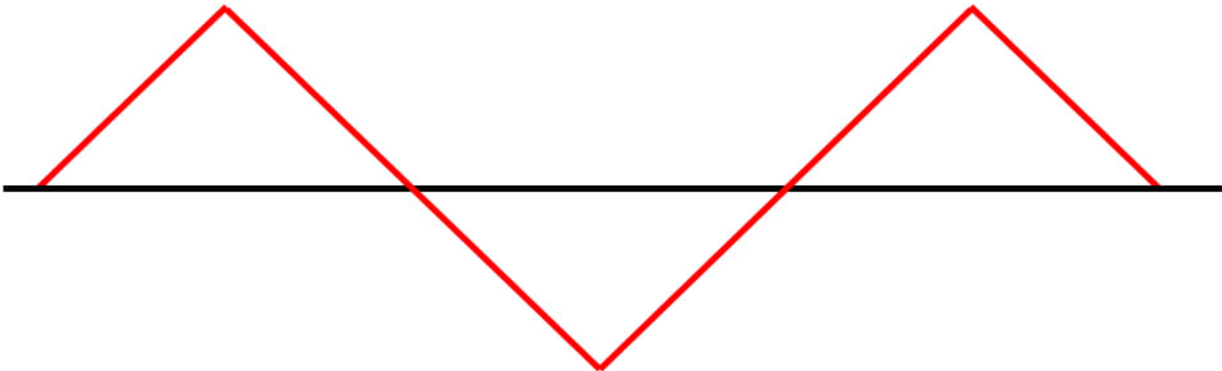
Satır 70'de 33 olan nota başlatma sayısını 17, satır 90'da 32 olan nota durdurma sayısını ise 16 yapın. Programınızın yeni hali şöyle olmalıdır:

ÖRNEK PROGRAM 3 (ÖRNEK PROGRAM 1'İN YENİ HALİ):

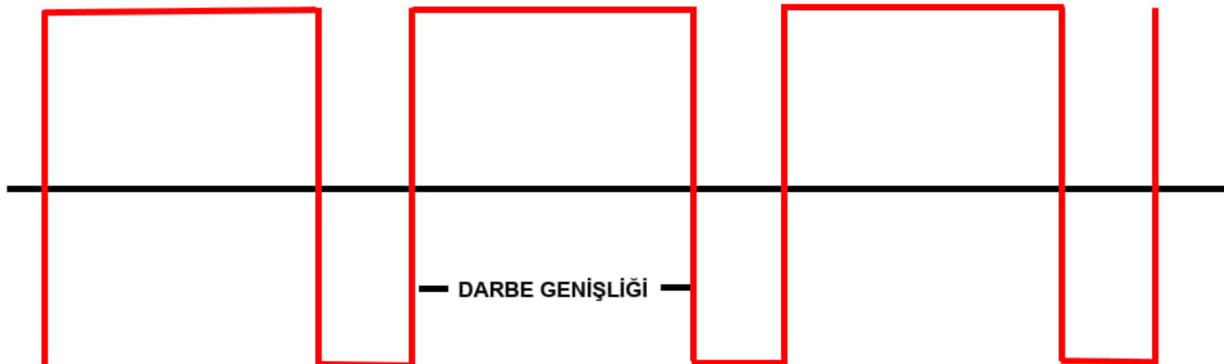
```
5 S=54272
10 FORL=STOS+24:POKEL,0:NEXT
20 POKES+5,9:POKES+6,0
30 POKES+24,15
40 READHF,LF,DR
50 IFHF<0THENEND
60 POKES+1,HF:POKES,LF
70 POKES+4,17
80 FORT=1TODR:NEXT
90 POKES+4,16:FORT=1TO50:NEXT
100 GOTO40
110 DATA25,177,250,28,214,250
120 DATA25,177,250,25,177,250
130 DATA25,177,125,28,214,125
140 DATA32,94,750,25,177,250
150 DATA28,214,250,19,63,250
160 DATA19,63,250,19,63,250
170 DATA21,154,63,24,63,63
180 DATA25,177,250,24,63,125
190 DATA19,63,250,-1,-1,-1
```

Şimdi programı çalıştırın (RUN).

Dikkat ederseniz sesin niteliği oldukça farklı, daha az tıngırtılı fakat daha tok ve derin. Bunun nedeni dişli dalga biçiminden üçgen dalga biçimine (aşağıdaki gibi) geçmemiz.



Üçüncü dalga biçimi ise değişken darbe (kare) dalgaları olarak adlandırılır.



Bu dikdörtgen biçiminde bir dalgadır ve darbe devrinin uzunluğu, yüksek dalga oranı tarafından belirlenir. Ses-1 için 2 ve 3 no.'lu kayıtlar bu işlevi yerine getirirler. Kayıt 2 darbe genişliğinin alt baytıdır (L_{pw} = 0-255 arası). Kayıt 3 ise, üst 4 bittir (H_{pw} = 0-15 arası).

Bu kayıtların hepsi birlikte 1 darbe genişliği için aşağıdaki formülü kullanarak hesaplayacağınız 12 bitlik bir sayıyı belirlerler.

$PW_n = H_{pw} * 256 + L_{pw}$

Darbe genişliği şu eşitlikle saptanır:

$PW_{out} = (PW_n / 40,95)\%$

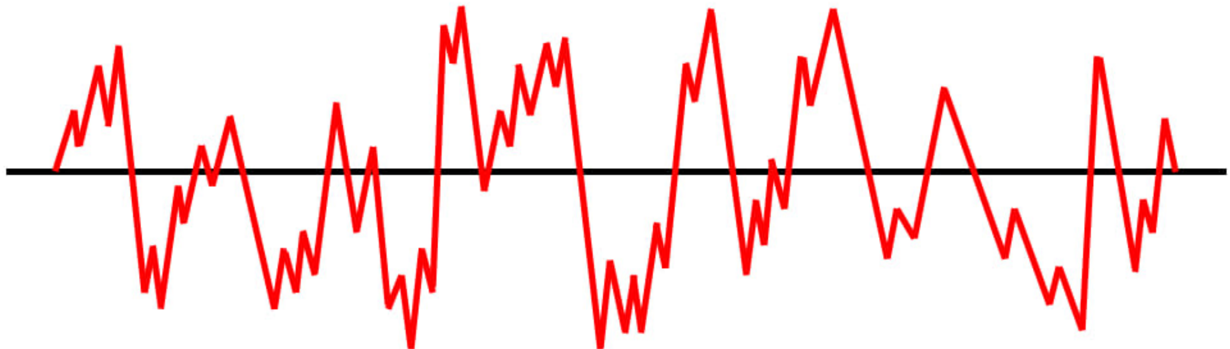
PW_n'nin değeri 2048 ise kare biçimi bir dalga elde edersiniz. Bu, kayıt 2 (L_{pw}) * 0, kayıt 3 (H_{pw}) = 8 demektir.

Şimdi programa şu satırı eklemeyi deneyin:

```
15 POKES + 3,8:POKES + 2,0
```

Daha sonra 70'inci satırda bulunan başlangıç sayısını 65, 90'ıncı satırda bulunan durdurma sayısını ise 64 olarak değiştirin ve programı çalıştırın (RUN). Şimdi de yüksek darbe genişliğini (15'inci satırdaki kayıt 3) 8 yerine 1 yapın. Gördüğünüz gibi, sesin niteliğinde çok dramatik bir değişim oldu.

Elde edebileceğiniz son dalga biçimi ise "beyaz gürültü" 'dür (Aşağıdaki şekil).

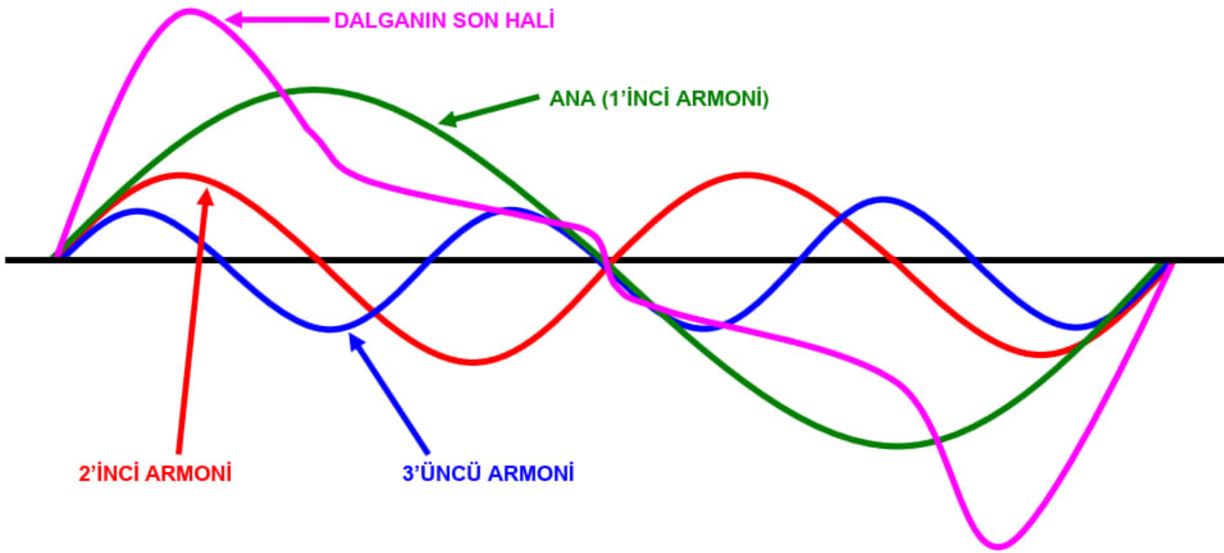


Bu genellikle ses efektleri için kullanılır. Elde edilecek sesi duymak isterseniz, 70'inci satırdaki başlangıç sayısını 129, 90'ıncı satırdaki bitiş sayısını da 128 olarak değiştirin.

DALGA BİÇİMLERİNİ ANLAMA:

Bir nota çalınırken, temel frekans ve harmonisinde salınan sinüs dalgalarından oluşur.

Temel frekans bir notanın tüm pitch'ini tanımlar. Harmonik, temel frekansın tamsayı katları olan frekanslara sahip sinüs dalgalarıdır. Ses dalgası onu oluşturan temel frekans ve tüm harmoniklerin toplamıdır.



Müzik kuramında temel frekans 1 numaralı harmoniktir. İkinci harmonik temel frekansın 2 katı, üçüncü harmonik ise temel frekansın 3 katı frekansa sahiptir. Bir notada bulunan harmonik miktarları o notanın timbre'ini belirlerler.

Gitar ya da keman gibi akustik enstrümanların çok karmaşık harmonik yapıları vardır. Gerçekte harmonik yapı bir notayı çalarken bile değişebilir. Artık Commodore 64 bilgisayarınızın müzik synthesizer'ında bulunan dalga biçimlerini kullandığınıza göre harmoniklerin nasıl testere dişli, üçgen ve dikdörtgen dalga biçimleriyle birlikte işlediği üzerine konuşabiliriz.

Bir üçgen dalga yalnızca tek sayılı harmoniklerden oluşur. Her harmoniğin miktarı, harmonik sayısının karesinin tersiyle orantılıdır. Yani harmonik 3, harmonik 1'den 1/9 kadar daha hafiftir, çünkü harmonik 3'ün karesi 9 ("3"), 9'un tersi ise 1/9'dur.

Sizin de görebileceğiniz gibi bir üçgen dalganın şekliyle temel frekansta salınan bir sinüs dalgasının şekli oldukça benzerdir.

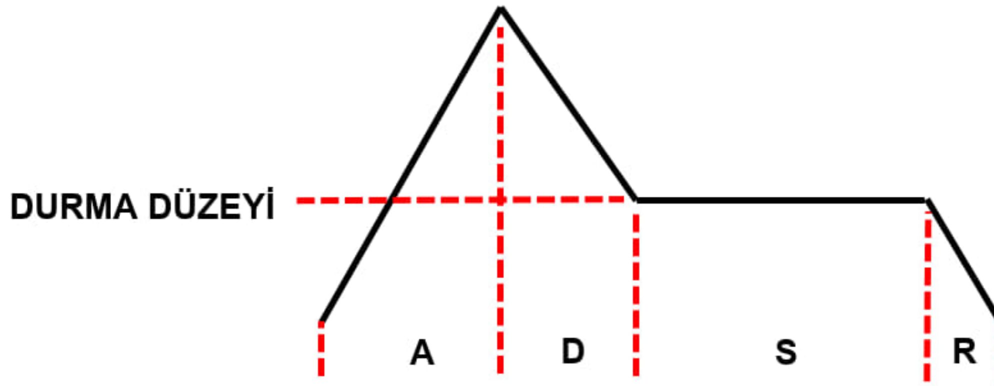
Dişli dalgalar tüm harmonikleri kapsarlar. Yer alan harmonik miktarları harmonik numarasının tersiyle orantılıdır. Örneğin harmonik 2, 1'den 1/2 kat daha yüksektir.

Kare dalgalar harmonik numarasının tersiyle orantılı miktarda tek harmonikleri içerirler. Diğer dikdörtgen dalgaların harmonik yapıları değişkendir. Bir dikdörtgen dalga sesinin timbre'i, sadece darbe genişliğinin değiştirilmesiyle oldukça farklılaşır.

Kullanacağınız dalga biçimini dikkatle seçerek, istediğiniz sese benzeyecek harmonik yapıya girebilirsiniz. Daha temiz, işlenmiş bir ses elde etmek istiyorsanız Commodore 64'ünüzün ses niteliği ile ilgili yönlerinden biri olan filtrelemeyi de kullanabilirsiniz. Bu konuya daha sonra gireceğiz.

ZARF ÜRETİCİSİ

Duyacağınız müzikal tonun ses düzeyi, ilk duyduğunuz andan itibaren sürekli değişir ve sonunda hiçbir şey duymaz olursunuz. Bir nota ilk çalınmaya başlandığı an ses, sıfır düzeyinden en üst düzeye doğru yükselmeye başlar. Bu değişim hızına yükselme (attack) adını veriyoruz. Daha sonra ise en üst düzeyden orta düzeye doğru düşmeye başlar. Notanın bu düşüş hızına ise düşme (decay) diyoruz. Orta düzey sesin kendisi ise durma (sustain) olarak adlandırılır. Sonunda nota durur ve sustain düzeyinden sıfır ses düzeyine doğru düşmeye başlar. Bu düşüş hızı ise kaybolma (release) olarak adlandırılır. Bu 4 evrelik değişimin bir çizimini veriyoruz.



Yukarıda sözünü ettiğimiz tüm bu değişimler bir notanın niteliklerini ve kısıtlamalarını belirlerler. Bu sınırlara parametreler diyoruz.

YÜKSELME / DÜŞME / DURMA / KAYBOLMA (ATTACK, DECAY, SUSTAIN, RELEASE), parametrelerinin tümü birden ADSR olarak adlandırılır. Bu parametreleri, ses üretici çipte bulunan diğer bir yerleşim seti ile kontrol edebilirsiniz. İlk örnek programı tekrar yükleyin (LOAD) ve çalıştırın (RUN). Sesi hatırlamış olmalısınız. Şimdi programın 20 sayılı satırını değiştirin.

ÖRNEK PROGRAM 4 (ÖRNEK 1'İN YENİ HALİ):

```
5 S=54272
10 FORL=STOS+24:POKEL,0:NEXT
20 POKES+5,88:POKES+6,195
30 POKES+24,15
40 READHF,LF,DR
50 IFHF<0THENEND
60 POKES+1,HF:POKES,LF
70 POKES+4,33
80 FORT=1TODR:NEXT
90 POKES+4,32:FORT=1TO50:NEXT
100 GOTO40
110 DATA25,177,250,28,214,250
120 DATA25,177,250,25,177,250
130 DATA25,177,125,28,214,125
140 DATA32,94,750,25,177,250
150 DATA28,214,250,19,63,250
160 DATA19,63,250,19,63,250
170 DATA21,154,63,24,63,63
180 DATA25,177,250,24,63,125
190 DATA19,63,250,-1,-1,-1
```

Kayıt 5 ve 6, ses 1 için ADSR niteliklerini tanımlar. Yükselme**,** kayıt 5'in üst nybble'ıdır. 1 nybble, yarım bayta eşittir. Yani; her kaydın 4 üst veya 4 alt açık/kapalı yerleşimleri (bitleri) 1 nybble'ı oluştururlar. Düşme alt nybble'dır. Yükselme için 0-15 arası herhangi bir sayı kullanmanız mümkündür. Düşme içinse. Seçtiğiniz bu sayıyı 16 ile çarpın ve yine 0-15 arası bir sayıya ekleyin. Bu sayılara karşılık gelen değerlerin bir listesi aşağıda verilmiştir.

Durma düzeyi ise kayıt 6'nın üst nybble'ı tarafından tanımlanır. Yine 0-15 arası bir sayı verebilirsiniz. Bu sayı, Durma düzeyinin erişeceği en yüksek ses düzeyinin oranını tanımlar. Kaybolma hızı ise kayıt 6'nın alt nybble'ı tarafından tanımlanır. (ADSA, parametrelerin İngilizce karşılıklarının baş harflerinden oluşmuştur.)

Aşağıda ADSR değerlerinin anlamlarını veriyoruz.

DEĞER	YÜKSELME ORANI (ZAMAN/DÖNGÜ)	DÜŞME/KAYBOLMA ORANI (ZAMAN/DÖNGÜ)
0	2 ms	6 ms
1	8 ms	24 ms
2	16 ms	48 ms
3	24 ms	72 ms
4	38 ms	114 ms
5	56 ms	168 ms
6	68 ms	204 ms
7	80 ms	240 ms
8	100 ms	300 ms
9	250 ms	750 ms
10	500 ms	1.5 s
11	800 ms	2.4 s
12	1 s	3 s
13	3 s	9 s
14	5 s	15 s
15	8 s	24 s

Bunlar örnek programınızda deneyebileceğiniz birkaç değer. Bunları ve kendi tercihlerinizi bir bir deneyin. Elde edeceğiniz ses çeşitlemeleri müthiş olmalı! Keman tipi bir ses elde etmek istiyorsanız satır 20'yi şöyle değiştirin:

```
20 POKES + 5,88;POKES + 6,89:REM A = 5;0
= 8;5 = 5;R = 9
```

Aşağıda verdiğimiz satırları girerseniz dalga biçimini üçgene çevirmiş olacaksınız. Bu da kselafon sesi elde etmenizi sağlayacaktır:

```
20 POKES+5,9:POKES+6,9:REM A= 0;D = 9;S
= 0;R = 9
70 POKES+4,17
90 POKES+4,16:FORT=1TO50:NEXT
```

Şu satırları girmeniz ise kare dalga biçimine geçmenizi, böylece de piyano sesi elde etmenizi sağlayacak:

```
15 POKES+3,8:POKES+2,0
20 POKES+5,9:POKES+6,0:REM A = 0;D = 9;S
 = 0;R = 0
70 POKES+4,65
90 POKES+4,64:FORT= 1TO50:NEXT
```

Herhalde en heyecan verici ses de müzik synthesizer'inin kendi sesi olacak. Şu satırı deneyin:

```
20 POKES+5,144:POKES + 6,243:REM A = 9;D
 = 0;S = 15;R = 3
```

FİLTRELEME

Bir dalga biçiminin harmonik yapısını bir filtre kullanarak değiştirebilirsiniz. SID çipi 3 tip filtreleme olanağı verir. Bunlar tek tek kullanılabileceği gibi bir kombinasyon içerisinde de kullanılabilir. Şimdi örnek programımıza geri dönelim ve bunu bir kez de basit bir filtre kullanarak deneyelim. Kullanabileceğimiz çok çeşitli filtre kontrolleri var.

Satır 15'i programınıza eklediğinizde filtrenin cutoff frekansı filtrenizin başlama (referans) noktasıdır. Kayıt-21'e yüksek frekans cutoff noktasını, kayıt-22'ye ise alçak frekans cutoff noktasını yerleştirir. Ses-1 için filtre açmak istiyorsanız kayıt-23'ü POKE'layın.

Daha sonra satır 30'u değiştirin. Burada yüksek-geçiş filtresi kullanacaksınız (Bak. SID kayıt haritası).

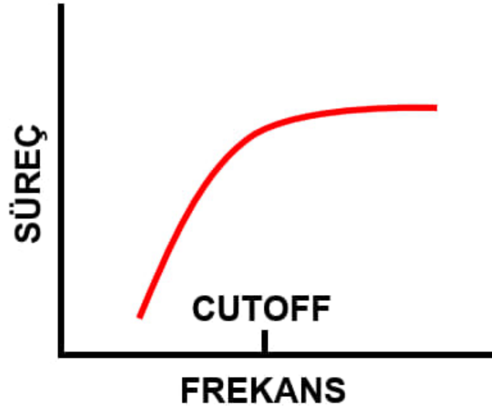
ÖRNEK PROGRAM 5 (ÖRNEK 1'iN YENİ HALİ):

```
5 S=54272
10 FORL=STOS+24:POKEL,0:NEXT
15 POKES+22,128:POKES+21,0:POKES+23,1
20 POKES+5,9:POKES+6,0
30 POKES+24,79
40 READ HF,LF,DR
50 IF HF<0 THEN END
60 POKES+1,HF:POKES,LF
70 POKES+4,33
80 FORT= 1 TO DR:NEXT
```

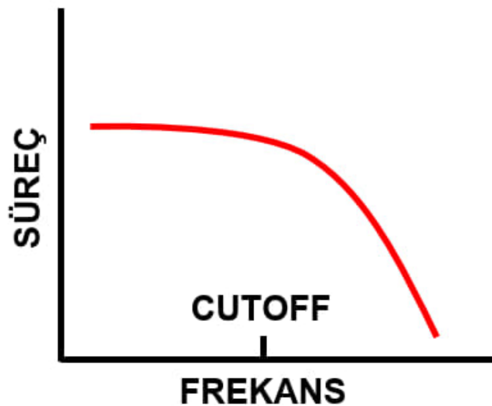
```
90 POKES+4,32:FORT=1 TO 50:NEXT
100 GOTO 40
110 DATA 25,177,250,28,214,250
120 DATA 25,177,250,25,177,250
130 DATA 25,177,125,28,214,125
140 DATA 32,94,750,25,177,250
150 DATA 28,214,250,19,63,250
160 DATA 19,63,250,19,63,250
170 DATA 21,154,63,24,63,63
180 DATA 25,177,250,24,63,125
190 DATA 19,63,250,-1,-1,-1
```

Şimdi programı çalıştırın (RUN). Dikkat ederseniz alçak tonların ses düzeyi düştü. Bu, nota sesinin bütününde bir değişikliğe yol açacaktır. Sesin niteliği şimdi daha teneke sesi gibi. Bunun nedeni belirli cutoff frekansının altındaki frekansları kısan yüksek geçiş filtresi kullanmanızdır.

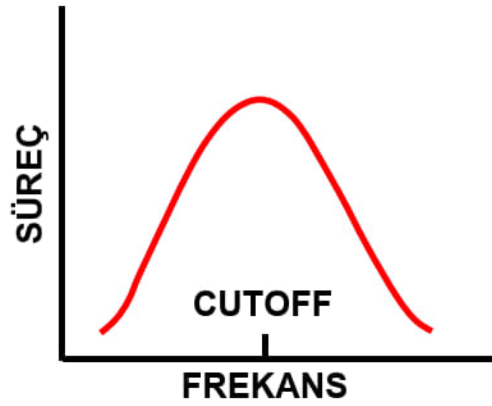
Commodore bilgisayarınızın SID çipinde 3 çeşit filtre vardır. Biz sadece yüksek-geçiş filtresinden yararlandık.



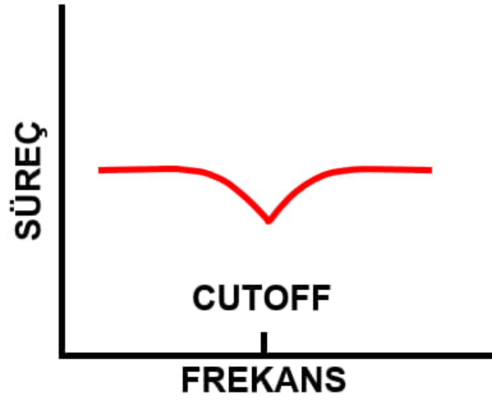
SID çipinde bulunan bir başka filtre çeşidi ise bandpass (bant geçişi) filtresidir. Bu filtre cutoff çevresinde çok dar bir alanda yer alan frekansları süzer (geçirir), diğerleri ise kısalır.



Sonuçta, çip bir bandpass filtresi ile donatılmıştır. Bu cutoff çevresinde çok dar bir alanda yer alan frekansları geçirir.



Yüksek ve alçak geçişli filtreler bir "notch reject filter" oluşturmak üzere birleştirilebilirler. Su, filtre cutoff frekansında seslerin düzeyini azaltırken, cutoff düzeyinden yüksek ya da alçak olan frekansları geçirir.



Kayıt 24, tüm ses düzeyi (volüm) kontrollerinin yanı sıra kullanmak istediğiniz filtre tipini de belirler. Bit 6 yüksek geçişli filtreyi (0 = Kapalı, 1 = Açık), bit 5 bandpass filtreyi, bit 4 ise alçak geçişli filtreyi kontrol eder. Cutoff frekansının alt üç biti kayıt 21 tarafından saptanır (L_{cf}) (L_{cf} = 0'dan 7'ye kadar). Yüksek cutoff frekansının 8 bitini ise kayıt 22 saptar. (H_{cf}) (H_{cf} = 0-255).

Filtreleme işlevini dikkatle kullanırsanız, bir dalga biçiminin harmonik yapısını istediğiniz gibi değiştirebilirsiniz. Böylece istediğiniz sesi elde etmeniz de kolaylaşacaktır. Bunun yanı sıra ADSA evrelerinde geçerken, bir sesin filtrelemesini değiştirerek ilginç efektler elde etmeniz de mümkündür.

İLERİ TEKNİKLER:

SID çipinin parametreleri bir ses ya da nota sırasında dinamik olarak değiştirilebilir. Böylece birçok ilginç ve eğlenceli efektler yaratabilirsiniz. Bunu kolaylaştırmak üzere kayıt 27'de yer alan osilatör 3 ve kayıt 28'de yer alan envelope üretici 3'ün dijital çıktıları kullanılır.

Kayıt 21'de yer alan osilatör 3'ün çıktısı, seçilen dalga biçimiyle direkt olarak ilişkilidir. Eğer osilatör 3'ün dişli dalgaları ile çalışmayı seçmişseniz, bu kayıt osilatör 3'ün frekansı tarafından belirlenen 0 ile 255 arası artımlı (derece, derece artan) sayılar sunacaktır. Eğer üçgen dalga biçimini seçmişseniz çıktı, önce basamak basamak 0'dan 255'e yükselecek, daha sonra ise 255'ten 0'a düşecektir. Eğer darbe dalgalarını seçmişseniz bu kez çıktı yine 0-255 arasında öne arkaya zıplayacaktır. Son olarak: gürültü dalga biçimi seçimi size bir seri rasgele sayı verecektir. Eğer osilatör 3'ü modülasyon (birimleme) amacıyla kullanıyorsanız genelde çıktıyı duymak istemezsiniz. Kayıt 24'ün 7'nci bitini bir yaparak (set) ses 3'ün işitsel çıktısını kapatabilirsiniz. Kayıt 27 daima, osilatör'ün değişen çıktısını yansıtır ve envelope (ADSR) üreticisinden asla etkilenmez.

Kayıt 25, osilatör 3'ün envelope üreticisinin çıktılarına erişmenizi sağlar. Kayıt 25 hemen hemen osilatör 3'ün çıktısı gibi çalışır. Bu kayıtlardan herhangi bir çıktı elde etmek istiyorsanız, osilatörü işler duruma getirmeniz gerekir.

Vibrato (frekansta hızlı bir değişim), osilatör 3'ün çıktısının diğer osilatörün frekansına eklenmesiyle elde edilebilir. Örnek program 6 hu işleyişi göstermektedir.

ÖRNEK PROGRAM 6:

```
10 S=54272
20 FORL=0 TO 24:POKES+L,0:NEXT
30 POKES+3,8
40 POKES+5,41:POKES+6,89
50 POKES+14,117
60 POKES+18,16
70 POKES+24,143
80 READ FR,DR
90 IF FR=0 THEN END
100 POKES+4,65
110 FORT=1 TO DR*2
120 FQ=FR+PEEK(S+27)/2
130 HF=INT(FQ/256):LF=LQ AND 255
140 POKES+0,LF:POKES+1,HF
150 NEXT
160 POKES+4,64
170 GOTO 80
500 DATA 4817,2,5103,2,5407,2
510 DATA 8583,4,5407,2,8583,4
520 DATA 5407,4,8583,12,9634,2
530 DATA 10207,2,10814,2,8583,2
540 DATA 9634,4,10814,2,8583,2
550 DATA 8583,12
560 DATA 0,0
```

Aşağıda örnek program 6'nın satır satır açıklamasını veriyoruz. Her satır ayrıntılı bir şekilde anlatıldığından programın işleyişini daha iyi kavrayacaksınız.

SATIRLAR	AÇIKLAMA
10	S değişkenine ses çipinin başlangıç adresini yerleştirilir.
20	Tüm ses çipi hanelerini temizler.
30	Ses 1 için yüksek darbe genişliğini yerleştirir.
40	Ses 1 için yükselme/düşmeyi yerleştirir (A=2, D=9).
	Ses 1 için durma/kaybolmayı yerleştirir (S=5, R=9).
50	Ses 3 için alçak frekansı yerleştirir
60	Ses 3 için üçgen dalga biçimini yerleştirir.
70	Ses (volüm) değerini 15 yapar, ses 3'ün işitsel çıktısını kapatır.
80	Notanın frekansını ve süresini (duration) okur.
90	Frekans 0 ise durur.
100	Ses 1 için başlangıç darbe dalga biçimi kontrolünü POKE eder.
110	Süre için zamanlama döngüsünü başlatır.
120	Sarkaç 3 çıktısını kullanarak yeni frekansı alır.
130	Yüksek ve alçak frekansları alır.
140	Ses 1 için yüksek ve alçak frekansları POKE eder.
150	Zamanlama döngüsünün sonu.
160	Ses 1 durma darbe dalga biçim kontrolünü POKE eder.
170	Sonraki nota için geri döner.
500-550	Şarkının frekans ve süreleri.
560	Sıfırlar şarkının sonunu işaret ediyor.

Dinamik efektler kullanarak da çok çeşitli ses efektleri yaratılabilir. Örneğin: aşağıdaki siren programı osilatör 3'ün üçgen dalgalarını temel aldığı koşullarda osilatör 1'in frekans çıktılarını dinamik olarak değiştirir.

ÖRNEK PROGRAM 7:

```
10 S=54272
20 FORL=0 TO 24:POKES+L,0:NEXT
30 POKES+14,5
40 POKES+18,16
50 POKES+3,1
60 POKES+24,143
70 POKES+6,240
80 POKES+4,65
90 FR=5389
100 FORT=1 TO 200
110 FQ=FR+PEEK(S+27)*3.5
120 HF=INT(FQ/256):LF=FQ-HF*256
130 POKES+0,LF:POKES+1,HF
140 NEXT
150 POKES+24,0
```

Aşağıda örnek program 7'nin satır satır açıklamasını veriyoruz. Her satır ayrıntılı bir şekilde anlatıldığından programın işleyişini daha iyi kavrayacaksınız.

ÖRNEK PROGRAM 7'NİN SATIR SATIR AÇIKLAMASI:

SATIRLAR	AÇIKLAMA
10	S değişkenine ses çipinin başlangıç adresini yerleştirilir.
20	Tüm ses çipi kayıtlarını temizler.
30	Ses 3 için alçak frekansı yerleştirir.
40	Ses 3 için üçgen dalga biçimi yerleştirir.
50	Ses 1 için yüksek darbe genişliği yerleştirir.
60	Ses yüksekliğini 15 yapar, ses 3'ün işitsel çıktısını kapatır.
70	Ses 1 için yükselme/düşmeyi yerleştirir (S = 15, R = 0).
80	Ses 3 için üçgen dalga kontrolü POKE başlangıcı.
90	Siren için gereken en alçak frekansı yerleştirir.
100	Zamanlama döngüsünü başlatır.
110	Sarkaç 3'ün çıktısını kullanarak yeni frekansı alır.
120	Alçak ve yüksek frekansları alır.
130	Ses 1 için yüksek ve alçak frekansları alır.
140	Zamanlama döngüsünü bitirir.
150	Sesi kapatır.

Gürültü dalga-biçimi geniş ölçekli ses efektleri yaratmak amacıyla kullanılır. Aşağıdaki örnek program filtrelenmiş gürültü dalga biçimini kullanarak alkış sesi çıkartır.

ÖRNEK PROGRAM:8

```
10 S=54272
20 FORL=0TO24:POKES+L,0:NEXT
30 POKES+0,240:POKES+1,33
40 POKES+5,8
50 POKES+22,104
60 POKES+23,1
70 POKES+24,79
80 FORN=1TO15
90 POKES+4,129
100 FORT=1TO250:NEXT:POKES+4,128
110 FORT=1TO30:NEXT:NEXT
120 POKES+24,0
```

Aşağıda örnek program 8'in satır satır açıklamasını veriyoruz. Her satır ayrıntılı bir şekilde anlatıldığından programın işleyişini daha iyi kavrayacaksınız.

ÖRNEK PROGRAM 8'İN SATIR SATIR AÇIKLAMASI:

SATIRLAR	AÇIKLAMA
10	S değişkenine ses çipinin başlangıç adresini yerleştirilir.
20	Tüm ses çipi kayıtlarını temizler.
30	Ses 1 için yüksek ve alçak frekanslar yerleştirilir.
40	Ses 1 için yükselme/düşmeyi yerleştirir (A = 0, D = 8).
50	Filtre için yüksek cutoff frekansını yerleştirir.
60	Ses 1 için filtreyi açar.
70	Ses yüksekliğini 15 yapar. Yüksek geçiş filtresini açar.
80	15 alkış sayar.
90	Başlangıç gürültü dalga biçimi kontrolünü yerleştirir.
100	Bekler, sonra bitiş gürültü dalga biçimi kontrolünü yerleştirir.
110	Bekler, sonra bir sonraki alkışa geçer.
120	Sesi kapatır.

SENKRONİZASYON (eş zamanlama) VE RİNG MODÜLASYONU

6581 SID çipinin bir özelliği de iki sesin senkronizasyonunu ya da ring modülasyonunu kullanarak oldukça karmaşık harmani yapıları oluşturmanızı sağlamasıdır.

Senkronizasyon temelde iki dalga biçiminin mantıksal VE'lenmesidir (AND). Eğer ikisi de 0'sa, çıktı 0'dır. Aşağıdaki örnek programla bir sivrisineğin sesini oluşturabilirsiniz.

ÖRNEK PROGRAM 9:

```
10 S=54272
20 FORL=0TO24:POKES+L,0:NEXT
30 POKES+1,100
40 POKES+5,219
50 POKES+15,28
60 POKES+24,15
70 POKES+4,19
80 FORT=1TO5000:NEXT
90 POKES+4,18
100 FORT=1TO1000:NEXT:POKES+24,0
```

Aşağıda örnek program 9'un satır satır açıklaması yer almaktadır.

ÖRNEK PROGRAM 9'UN SATIR SATIR AÇIKLAMASI:

SATIRLAR	AÇIKLAMA
10	S değişkenine ses çipinin başlangıç adresini yerleştirilir.
20	Tüm ses çipi kayıtlarını temizler.
30	Ses 1 için yüksek frekans yerleştirilir.
40	Ses 1 için yükselme/düşmeyi yerleştirir (A = 13, D = 11).
50	Ses 3 için yüksek frekans yerleştirilir.
60	Ses yüksekliğini 15 yapar.
70	Ses 1 için sync dalga biçimi kontrolü, üçgen başlangıcını tanımlama.
80	Zamanlama döngüsü.
90	Ses 1 için sync dalga biçimi kontrolü, üçgen bitimini tanımlama.
100	Ses kapanıncaya kadar bekler.

Senkronizasyon özelliği 70 sayılı satır da devreye girer. Kayıt 4'ün 0-1 ve 4 sayılı bitleri kullanılır. Bit 1, ses 1 ve ses 3'ün senkronizasyonunu sağlar. Bit 0 ve 4 ise her zamanki işlevlerini; yani sesi alır ve üçgen dalga biçimine ayarlar.

Ring modülasyonu (aşağıdaki programın 70 sayılı satırında ses 1 için, kayıt 4'ün 3'üncü biti 1 yapılarak oluşturuluyor), osilatör 1'in üçgen çıktısını, osilatör 1 ve osilatör 3'ün ring modulated kombinasyonu ile değiştirir. Bu zil ya da gang sesini andıran, armonik olmayan yüksek tonda yapılar üretir.

ÖRNEK PROGRAM: 10

```
10 S=54272
20 FORL=0TO24:POKES+L,0:NEXT
30 POKES+1,130
40 POKES+5,9
50 POKES+15,30
60 POKES+24,15
70 FORL=1TO12:POKES+4,21
80 FORT=1TO1000:NEXT:POKES+4,20
90 FORT=1TO1000:NEXT:NEXT
```

Aşağıda örnek program 10'un satır satır açıklaması yer almaktadır.

ÖRNEK PROGRAM 10'UN SATIR SATIR AÇIKLAMASI:

SATIRLAR	AÇIKLAMA
10	S değişkenine ses çipinin başlangıç adresini yerleştirilir.
20	Tüm ses çipi kayıtlarını temizler.
30	Ses 1 için yüksek frekans yerleştirilir.
40	Ses 1 için yükselme/düşmeyi yerleştirir (A = 0, D = 9).
50	Ses 3 için yüksek frekans yerleştirilir.
60	Ses yüksekliğini 15 yapar.
70	Dinglerin sayısı, üçgen başlangıcı, ses 1 için halka modu dalga biçimi kontrolü.
80	Zamanlama döngüsü, üçgen bitimi ve halka modu tanımı.
90	Zamanlama döngüsü, bir sonraki ding.

Commodore 64'ünüzün SID çipi parametrelerini kullanarak oluşturacağınız efekt sayısı çok çeşitlidir. Sadece kendi kendinize yapacağınız denemelerle bilgisayarınızın yeteneklerini keşfederek mutlu olacaksınız. Bu el kitabında verdiğimiz örnekler sadece çölde bir kum tanesidir.

BÖLÜM 5

BASIC'TEN MAKİNE DİLİNE

- Makine Dili Nedir?
- Makine Dili Programları Nasıl Yazabilirsiniz?
- Onaltılı Sayalar
- Adresleme Çeşitleri
- İndeksli Adresleme
- Altprogramlar
- Yeni başlayanlar İçin Yararlı İpuçları
- Büyük Bir İşe Başlarken
- MCS6510 Mikroişlemcisinin Komut Kümesi
- Commodore 64'ün Bellek Yönetimi
- KERNAL'ın Başlangıç Aktiviteleri
- BASIC'ten Makine Dilini Kullanmak
- Commodore 64 Bellek Haritası

MAKİNE DİLİ NEDİR?

Her mikrobilgisayarın içinde bir merkezi mikroişlemci vardır. Bu işlemci çok özel bir çiptir ve bilgisayarın beynidir. Her mikroişlemci kendi dilinin komutlarını anlar. Bu komutlara, makine dili komutları denir. Daha açık bir anlatımla makine dili, Commodore 64'ün anladığı tek programlama dilidir, yani bilgisayarınızın ana dilidir.

Eğer makine dili, Commodore 64'ün anladığı tek dil ise, CBM BASIC programlama dilini nasıl anlayabiliyor diye bir soru aklınıza takılabilir. CBM BASIC Commodore 64'ün makine dili değildir. Peki, o zaman Commodore 64, PRINT ve GOTO gibi CBM BASIC komutlarını nasıl anlar?

Bu soruya cevap verebilmek için, ilk önce Commodore 64'ünüzün nasıl çalıştığını bilmeniz gerekir. Commodore 64'ünüzün beyni olan mikroişlemciden ayrı olarak, silinmez belleğine kaydedilmiş bir de makine dili programı vardır. Bu program, Commodore 64'ünuzün işletim sistemi olarak adlandırılır ve sizin yazabileceğiniz programlardan farklı olarak bilgisayarınız kapatıldığı zaman kaybolmaz. İşletim sistemi (programı) otomatik olarak çalıştığı için Commodore 64 açıldığı zaman ne yapacağını bilir.

İşletim sisteminin, cihazınızın içindeki belleğin tümünü çeşitli görevler için organize etmek yükümlülüğü vardır. Ayrıca diğer fonksiyonlara ek olarak, klavyeden yazdığınız karakterlerin ekrana çıkmalarını sağlar. Mikroişlemci eğer bilgisayarınızın "zekâsı" ise, işletim sistemi, Commodore 64'ünüzün "kişiliği" olarak düşünülebilir. Bilgisayarı açtığınız zaman kontrolü eline alır ve gerekeni yaptıktan sonra şöyle der:

```
READY.
█
```

(Hazırım)

İşletim sistemi daha sonra, klavyeyi ve içinde hazır bulunan ekran editörünü kullanmanızı sağlar. Gerçekte sistemin sadece bir parçası olan ekran editörü, size takipçiyi hareket ettirme, karakter silme (DELETE) ve araya karakter yerleştirme (INSERT) gibi işlemleri yapabilme olanağı sağlar.

CBM BASIC'te bulunan komutların tümü, Commodore 64'ün içinde bulunan makine dili ile yazılmış bir diğer program yardımıyla anlaşılabilir. Bu program, işlenecek BASIC komutuna bağlı olarak, makine dilinin uygun olan kısmını çatıştırır. Her komutu bir bir açıkladığından bu program, BASIC yorumlayıcı (INTERPRETER) olarak adlandırılmıştır. Anlamadığı bir komutla karşılaştığında ise ekranda şöyle bir hata uyarısı görülür:

```
?SYNTAX ERROR
READY.
█
```

(Yazılım hatası)

MAKİNE KODLARI NEYE BENZER?

CBM Basic dilinde, bellek yerleşimlerini değiştirmek için kullanılan PEEK ve POKE komutlarına alışık olmalısınız. Sanırız ki bu komutları grafik çizdirmek veya ses efektleri yapmak için kullandınız. Her bellek yerleşimi, kendisini tanımlayan bir numaraya sahiptir. Bu numara, bellek yerleşiminin adresi olarak bilinir. Eğer Commodore 64'ün belleğini bir caddedeki binalara benzetirsek, her evin kapısındaki numara onun adresi olacaktır. Şimdi caddenin ne tarafı, hangi amaçla kullanılıyor ona bakalım:

COMMODORE 64'ÜN BELLEK HARİTASI

ADRESLER		AÇIKLAMA
ONLU	ONALTILI	
0 ve 1	0000-0001	-6510 bellek Giriş/Çıkış kontrol kayıtları
2'den 1023'e kadar	0002-03FF	-İşletim sistemi ve BASIC yorumlayıcı tarafından kullanılan çalışma alanı
1024'den 2023'e kadar	0400-07E7	-Ekran belleği
2040'dan 2047'ye kadar	07F8-07FF	-Yaratık göstergeçleri
2048'den 40959'a kadar	0800-9FFF	-Sizin kullanabileceğiniz serbest bellek. BASIC ya da makine diliyle yazacağınız programlar burada saklanır.
40960'dan 49151'e kadar	A000-BFFF	-8K CBM BASIC Yorumlayıcı (silinmez bellek)
49152'den 53247'ye kadar	C000-CFFF	-Basic dilinde olmayan özel programlar yazabileceğiniz serbest bellek alanı
53248'den 53294'e kadar	D000-D02E	-VIC-II görüntü kontrol kayıtları
54272'den 55295'e kadar	D400-D41C	-SID ses kontrol kayıtları
55296'dan 56296'ya kadar	D800-DBE7	-Ekran renk belleği
56320'den 57343'e kadar	DC00-DFFF	-Giriş/Çıkış kontrol kayıtları
57344'den 65535'e kadar	E000-FFFF	-8K CBM KERNAL İşletim Sistemi (silinmez bellek)

Şimdilik, belleğin bölümlerinin ne işe yaradığını anlamadıysanız bile, elinizdeki kitabın daha sonraki bölümlerinde bu konu daha açık olarak anlatılacaktır.

Demek ki: Commodore 64'ünüzün toplam 65536 bellek adresi bulunmaktadır. Bu adreslerden her birine 0 ile 255 arasında bir değer (1 baytlık bir rakam) yerleştirmek mümkündür. Bu rakam yerine göre, bir program komut kodu (Örneğin: PRINT anlamına gelen 153 rakamı), bir metnin bir karakteri (örneğin; A anlamına gelen 65 rakamı) veya herhangi bir veri değeri olabilir. O anda işlemekte olan program (bu, makinenin belleğindeki silinmez bir program veya sizin yazdığınız makine dili bir program olabilir) bellekteki rakamları sıradan işleyerek, program kodu mu, metin karakteri mi yoksa başka bir veri mi olduğuna karar verir.

Makine dili komutları, her biri basit bir aritmetik veya mantıksal bir işlem yapan bir baytlık kodlardan ibarettir. Bir baytlık komut kodunu, komutun çeşidine göre ya başka bir komut kodu ya da bir veya iki baytlık bir parametre (değer) izler.

Basic'in aksine makine dili program ve rutinleri, belleğin serbest olan herhangi bir bölgesine yerleştirmek ve bellekte aynı anda birden fazla makine dili programı bulundurmak mümkündür.

6510 MİKROİŞLEMCİ İÇERİSİNDEKİ KAYITLAR:

Makine dili komutları oldukça ilkeldir. BASIC'te alıştığımız türden isimli değişkenler yoktur. Karmaşık işlemleri kolayca gerçekleştiren matematiksel ifadeler veya PRINT, INPUT gibi komutlar yoktur. Bu yüzden, tek bir komut fazla bir iş yapamaz. Komutların başlıca işlevleri:

1. Bir bellek adresinin değerini değiştirmek veya bir yerden bir yere taşımak.
2. Bir baytlık bir sayı üzerinde basit aritmetik ve mantık işlemleri yapmak.
3. Program akışı içinde bir noktadan diğerine dallanmaktır.

Bu işlemlerde temel görev, 6510 mikroişlemcisinin kayıtlarına düşer.

Kayıtları: çeşitli değerleri, bellek içinde bir yerden diğerine taşımaya veya çeşitli işlemler yapmaya yarayan bir çeşit "taşıyıcı" ya da "kutu" olarak düşünebiliriz,

Kayıt (Register) sözcüğü Türkçe'ye, bazen "yazmaç" bazen de "sicil" olarak çevrilmektedir.

AKÜMÜLATÖR

Mikroişlemci içindeki en önemli kayıttır. Çeşitli komutlar kullanarak, akümülatördeki bilgiyi bellekteki bir yerleşime ya da belleğin içeriğini akümülatöre aktarmak veya herhangi bir kaydın içeriğini değiştirmek mümkündür. Ayrıca, akümülatör aritmetiksel işlemler yapabilmenizi sağlayan komutlara izin veren tek kayıttır.

X İNDEKSİ (X Index Register)

Akümülatör ile yapabileceğiniz işlemler için belirli komutlar olduğu gibi, bazı işlemleri yalnızca X kaydının kullandığı komutlar ile yapabilirsiniz. Çeşitli makine dili komutları, belleğin içeriğini X'e, X'in içeriğini de belleğe aktarabilmenizi, ya da belirli bir kaydın içeriğini değiştirebilmenizi sağlar.

Y İNDEKSİ (Y Index Register)

Akümülatör veya X için belirli komutlar olduğu gibi, yalnızca Y ile kullanabileceğiniz komutlar vardır. Bu komutları kullanarak belleğin içeriğini Y indeksine, Y'nin içeriğini belleğe aktarabilmeniz ya da diğer bir kaydın içeriğini doğrudan değiştirmeniz mümkündür.

MİKROİŞLEMCİ STATÜ KAYDI (Status Register)

Bu kaydın ötekilerden çok farklı bir işlevi vardır. Statü kaydını oluşturan 8 bitten her biri, mikroişlemcinin o andaki durumunun çeşitli yönlerini gösteren bir evet/hayır göstergesi (Flag) olarak kullanılır.

Statü kaydının bitleri 1 ya da 0 olmalarına göre aşağıda yer alan durumlarla ilgili bilgi verirler:

N	V		B	D	I	Z	C
•	•	•	•	•	•	•	•
7	6	5	4	3	2	1	0

7) N <Negatif) Bir önceki işlem negatif mi?
6) V (Overflow: taşma) Bir önceki işlemin sonucu ±127'den büyük mü?
5) -boş- daima 1 değerini taşır.
4) B (Break) Program kesintiye uğramış mı?
3) D (Decimal: onlu) işlemci onlu işlem modunda mı?
2) I (Interrupt: kesinti) kesinti durdurulmuş mu?
1) Z (Zero: sıfır) Bir önceki işlemin sonucu sıfır mı?
0) C (Carry: kalan) Bir önceki toplama, çıkarma veya kıyaslama işleminde kalan var mı?

Örneğin: Statü kaydında onaltılı 32 (onlu 50) değeri, son işlemin sonucunun sıfır olduğunu, işlemcinin onlu modda olmadığını, kesintinin işlendiğini, önceki işlemin sonucunun negatif olmadığını, taşma ve kalan olmadığını ve programın kesintiye uğradığını gösterir.

PROGRAM SAYACI (Program Counter)

Bu sayaç, o anda işlenmekte olan makine dili komutunun bulunduğu adresi içerir. Commodore 64'ünüzün (veya herhangi bir bilgisayarın) içinde bulunan işletim sistemi sürekli çalıştığından, program sayacı sürekli değişir. Program sayacını durdurmak için mikroişlemciyi durdurmak şarttır.

YIĞIN GÖSTERGECİ (Stack Pointer)

Bu kayıt yığındaki ilk boş yerleşimin adresini gösterir. Yığın, makine dili ile yazılmış programlarda geçici saklama yeri olarak kullanılır. Ayrıca, programın RTS komutu ile karşılaştığında geri döneceği adres yığında saklanır.

GİRİŞ/ÇIKIŞ ALANI (Input/Output Port)

Bu kayıt, belleğin 0'ıncı (veri yönü kaydı için) ve 1'inci (veri alanı için) yerleşimlerinde yer alır. 8 bitlik giriş/çıkış alanıdır. Commodore 64'te bu kayıt, çipin RAM ve ROM belleğinin 64K'sından fazlasını kontrol edebilmesini sağlamak amacıyla Bellek Yönetimi (Memory Management) için kullanılır.

Bu bölümde, yukarıdaki kayıtlar hakkında ayrıntılı bilgiler verilmemiş, yalnızca prensipleri konusunda gerekli açıklamalar yapılmıştır.

MAKİNE DİLİ İLE NASIL PROGRAM YAZABİLİRSİNİZ?

Makine dili programlar belleğe yüklendiğinden ve bilgisayarınızda bu programları yazma ve düzeltme kabiliyeti olmadığından, bu işlemleri yapabilmek için ya ayrı bir program kullanmalı ya da kendiniz bir BASIC program yazmalısınız.

Makine dili ile programlar yazmak için genellikle assembler (çevirici) programları kullanılır. Bu programlar, makine dili programların komutlarının anlaşılmasında bir dizi sayı ile uğraşılması yerine, anlaşılması ve akılda kalması kolay "mnemonic" dediğimiz kısaltılmış komut isimlerinin kullanılmasını sağlar. Makine dili programları daha anlaşılabilir ("mnemonic") biçimlerde yazabilmenizi sağlayan programlara "Assembler" denir. Bu işlemin tersi olan, makine dilini "mnemonic" biçimlerde gösteren programlara da "Disassembler" denir. Makine dilindeki programları incelemenizi ve değiştirmenizi sağlayan ve genellikle küçük çaplı bir assembler içeren utility paketlerine ise "Monitör" adı verilir, (Ekran anlamına gelen monitör ile karıştırmayın!) Basit veya ileri düzeyde makine dili monitör kasetlerini, Commodore satıcılarında bulabilirsiniz.

64MON

Satıcınızdan sağlayabileceğiniz 64MON kaseti, sizin CBM BASIC dünyasından makine diline geçebilmenizi sağlar. Bu program, ekran editörünü kullanarak, 6510 mikroişlemcisinin iç kayıtlarının içeriğinin ve belleğin herhangi bir kısmının ekrana çıkmasını ve bunları değiştirebilmenizi sağlar. Ayrıca kendi assembler ve disassembler'ı ile birlikte sizin makine dili programlarınızı kolayca yazabilmenize ve düzeltmenize olanak verir. Programlarınızı yazarken çevirici kullanmak zorunda değilsiniz, fakat çevirici ile işiniz daha kolaylaşacaktır. Makine dili ile program yazabilmeniz için herhangi bir çevirici programı kullanmanız sizin yararınıza olacaktır. Çevirici olmaksızın, programınızı doğrudan belleğe “POKE” komutları ile yerleştireceksiniz, bu da bizim size tavsiye edebileceğimiz bir yöntem değil.

Bu el kitabındaki bundan sonra yer alacak örnekler, 64MON'un kullandığı formatlarda verilecektir. Yaklaşık olarak tüm çevirici formatları birbirine benzediğinden, burada vereceğimiz örnekler diğerleri için de yararlı olacaktır.

Şimdi 64MON"un diğer özelliklerinden bahsetmeden önce size, biraz da onaltılı sayılama sisteminden söz edelim.

ONALTILI SAYILAR

Onaltılık sistem, çoğu makine dili programcısının, bir sayı veya bir adresten söz ederken kullandığı bir yazım çeşididir.

Bazı çeviriciler sizin adresler ve sayılar için onaltılı sistemle birlikte, onlu (10 tabanı), ikili (2 tabanı) ve hatta sekizli (8 tabanı) sistemleri de kullanmanıza olanak verir. Bu çeviriciler bir sayı tabanından diğerine değişimleri kendileri yaparlar.

Onaltılı yazım, ilk başlarda size biraz zor gelse de yapacağınız alıştırmalarla kısa zamanda konuyu iyice öğrendiğinizi göreceksiniz.

Onluk (10 tabanına göre) sistemi incelerseniz sayıların 0 ile tabanın bir eksiği, yani 9 rakamından oluştuğunu fark edeceksiniz. Bu temel ilke, bütün sayı tabanları için geçerlidir. İkili sistemlerde de 0 ve 1 (tabandan bir eksik) rakamları kullanılır. Aynı şekilde 16'lı sistemde de rakamlar, 0 ile 15 arasında olmalıdır. Ama 10'dan 15'e kadar olan sayılar tek basamaklı olmadığından onların yerine alfabenin ilk altı harfi kullanılmıştır.

ONLU	ONALTILI	İKİLİ
0	0	00000000
1	1	00000001
2	2	00000010
3	3	00000011
4	4	00000100
5	5	00000101
6	6	00000110
7	7	00000111
8	8	00001000
9	9	00001001
10	A	00001010
11	B	00001011
12	C	00001100
13	D	00001101
14	E	00001110
15	F	00001111
16	10	00010000

Şimdi başka bir şekilde inceleyelim; aşağıda 10'lu sistemin yapısını gösteren bir örnek veriyoruz.

4569 sayısını ele alalım:

Çoğalan katları				
İle taban :	10^3	10^2	10^1	10^0
Eşittir :	1000	100	10	1
4569 (10 tabanına göre)	4	5	6	9
= (4 X 1000) + (5 X 100) + (6 X 10) + 9				

Bir de 16'lık sistemin yapısını gösteren örneğe bakalım.

Çoğalan katları				
İle taban :	16^3	16^2	16^1	16^0
Eşittir :	4096	256	16	1
11D9 (16 tabanı) için	1	1	D	9
= 1 X 4096 + 1 X 256 + 13 X 16 + 9				

Öyleyse 4569 (10 tabanı) = 11D9 (16 tabanı)

Daha önce de söylendiği gibi adreslenebilen bellek aralığı 0 ile 65535 arasındadır. Onaltılık sistemde bu aralık 0 ile FFFF'dir.

Onaltılık sistemde sayılar genellikle önlerindeki bir dolar işareti ($) ile gösterilirler. Bu işaret onaltılı sayıları, onlu sayılardan ayırmak için kullanılır. Şimdi 64MON'u kullanarak, belleğin bir kısmının içinde bulunan onaltılı sayıları görelim.

64MON'u yükleyip, iki SYS komutundan (SYS 8*4096 veya SYS 12*4096) birini işletiniz. Ekranda şunu göreceksiniz:

```
SYS 8*4096
B*
   PC   SR AC XR YR SP
.; 0401 32 04 5E 00 F6
```

Eğer aşağıdakileri yazıp,

```
.M 0000 0020
```

RETURN tuşuna basarsanız, 9 sıra onaltılı sayı göreceksiniz. Sıranın ilk 4 basamağı, belleğin ilk baytının adresini, sonraki 8 basamak da o adresteki bellek yerleşiminin içeriğini gösterir.

Onaltılı sistemde düşünmeyi öğrenmek için çaba sarf etmelisiniz. Tekrar onlu sisteme çevirmek gerekmediğinden, o kadar zor bir iş olmadığını göreceksiniz. Örneğin, herhangi bir değerin 5357 yerine $14ED'de saklandığını söylerseniz hiçbir şey fark etmeyecektir.

İLK MAKİNE DİLİ KOMUTUNUZ

LDA - AKÜMÜLATÖRE YÜKLE

6510 Assembler'inde, mnemonic kodlar hep 3 karakterlidir. LDA komutu "akümülatöre... yükle" anlamındadır ve ne yükleneceği komutla birlikte verilen değişkenle belirtilir. Assembler, hangi komutun hangi sayı koduyla gösterildiğini bilir ve bir komutu çevirdiğinde belleğin adresi önceden saptanmış bir bölümüne yerleştirir. 6510 mikroişlemcisine ya da assembler'a uymayan, çeviremedikleri sözcükler için hata mesajları ya da uyarılar verir.

Eğer koddan sonraki parametrenin önüne "#" işareti koyarsanız bu, ilgili kayda, "#" işaretinden sonraki sayının yüklenmesi gerektiği anlamına gelir. Örneğin:

LDA #$05 ··············· $ = ONALTILI

Bu komut, akümülatöre $05 (onluk sistemde 5) değerini yerleştirir. Çevirici, belirlenen ilk adrese $A9 (bu, LDA komutunun karşılığı olan makine dili koddur) bunu izleyen adrese $05 değerini yükleyecektir.

Komutu izleyen parametrenin önünde "#" işareti varsa, komut, "immediate" (dolaysız) moddadır: yani, akümülatöre yüklenen değer, izleyen parametrenin ta kendisidir.

Diyelim ki, akümülatöre bildiğiniz belli bir değeri değil de belli bir adreste o an için hangi değer varsa onu yüklemek istiyorsunuz. Kullanılacak olan komut, yine LDA (Akümülatöre... yükle) komutudur. Ancak, komutu izleyen parametre dolaysız bir değeri değil, bir adresi belirtecektir. Örneğin:

LDA $102E

Çevirici ikinci komuttaki değişkenin önünde "#" işareti olmadığından parametrenin dolaysız bir değer değil, bir adres olduğunu anlayabilir. 6510 mikroişlemcisi de aynı ayrımı, komut kodunun değişik olmasıyla yapar. LDA (immediate)'ın sayısal kodu $A9, LDA (absolute)'un sayısal kodu ise $AD'dir.

Makine dilindeki sayısal komut kodları, sadece, genelde komutun ne yapacağını (Akümülatöre bir değer yüklemek gibi) değil, bu işlemi tam olarak nasıl yapacağını ve komutu izleyen parametrenin nasıl yorumlanacağını da belirtirler. Örneğin: $A9, $AD, $A5, $B9, $B1, $BD, hepsi de LDA anlamına gelir. Fakat yukarıda da gördüğümüz gibi, $A9, komutu izleyen 1 baytlık değerin direkt olarak akümülatöre yükleneceğini, $AD ise komutu izleyen 2 bayt tarafından gösterilen adresteki değerin akümülatöre yükleneceğini belirtir.

Bir kodu temsil eden anımsatıcı "mnemonic", genellikle onun ne yaptığını ima eder. Örneğin, başka bir komut olan LDX'i ele alırsak, bunun ne işe yaradığını düşünüyorsunuz?

"X kaydını şununla yükle..." dediyseniz, sınıfın en üstüne gidin. Yapmadıysanız merak etmeyin, makine dilini öğrenmek sabır ister ve bir günde öğrenilemez.

Çeşitli dahili kayıtlar, bir bayt bilgi tutabildikleri için özel bellek konumları olarak düşünülebilir. İkili numaralandırma sistemini (2 tabanı) açıklamamıza gerek yok, çünkü daha önce ana hatlarıyla belirtilen onaltılık ve ondalık sayılarla aynı kurallara uyuyor, ancak bir "bit" bir ikili rakamdır ve sekiz bit bir baytı oluşturur! Bu, bir baytta bulunabilecek maksimum sayının, sekiz basamaklı bir ikili sayının olabileceği en

büyük sayı olduğu anlamına gelir. Bu sayı 11111111'dir (ikili), bu da $FF'ye (onaltılık) eşittir ve 255'e (ondalık) eşittir. Bir bellek konumuna neden yalnızca 0'dan 255'e kadar sayıların konulabileceğini muhtemelen merak etmişsinizdir. POKE 7680, 260'ı denerseniz ("iki yüz altmış sayısını yedi bin altı yüz seksen hafıza konumuna koyun" diyen bir BASIC ifadesidir), BASIC yorumlayıcısı yalnızca 0 ila 255 arasındaki sayıların, bir hafıza konumuna girilebileceğini bilir ve Commodore 64'ünüz şu şekilde cevap verecektir:

```
?ILLEGAL QUANTITY ERROR

READY.
█
```

Bir baytın alabileceği en yüksek değerin 255 (Onaltılı $FF) olduğunu biliyoruz. O halde, "mutlak" (absolute) komut "LDA $102E"'deki adres parametresi bellekte nasıl gösterilir? Şüphesiz, bir bayt ile gösterilemeyeceğinden, iki baytla gösterilir. En sağdaki iki basamak adresin "alt baytı" (low byte), soldaki iki basamak da "üst baytı" (high byte) olacaktır.

6510 mikroişlemcisinde, adreslerin ters sırayla, yani, önce alt bayt, sonra da üst bayt olarak yazılması gerektiğini öğrenmeniz gerekiyor. Buna göre "LDA $102E" komutu bellekte şu üç ardışık değeri alacaktır.

$AD, $2E, $10

Şimdi, ilk programınızı yapabilmeniz için bir komuta daha ihtiyacınız var. Bu komut da BRK'dır. Bunu şimdilik makine dilinin END komutu olarak düşünebilirsiniz.

64MON'la bir program yazar ve sonuna da BRK komutu koyarsak, program çalışıp bittiğinde tekrar 64MON'a döner. Buna karşılık, programınızda bir hata varsa veya BRK komutuna ulaşamıyorsa, program hiçbir zaman 64MON'a dönmeyecektir. **STOP** tuşu da sadece BASIC programlarda işlediğinden, programdan çıkıp düzeltmeler yapmanıza imkân yoktur. Bu da makine dilinde programlar yazmanın BASIC'e oranla en büyük zorluklarından birini oluşturur.

İLK PROGRAMINIZI YAZARKEN

Eğer ekrana karakter bastırmak için BASIC'teki POKE komutu kullandıysanız, POKE için karakter kodlarının CBM ASCII karakter değerlerinden ayrı olduğunun farkındasınızdır. Örneğin: aşağıdaki komutu girerseniz Commodore 64 size ekranda şöyle karşılık verir:

```
PRINT ASC("A")
 65

READY.
█
```

POKE komutunu kullanarak ekrana A harfini çıkartmak istiyorsanız, kullanacağınız kod 1'dir. Önce:

SHIFT **CLR/HOME** tuşlarına basarak ekranı temizleyin, daha sonra da:

```
POKE 1024,1:POKE 55296,14
```

yazın **RETURN** tuşuna basın. (1024 ekran belleğinin başlangıç adresidir.) POKE kelimesindeki "P" şimdi "A" olmalıdır.

Aynı işlemi bir de makine diliyle yapmayı deneyelim. 64MON'a aşağıdakileri yazın. (Takipçinin "." işaretinin yanında yanıp sönmesi gerekir.)

```
.A 1400 LDA #$01
.A 1402 STA $0400
.A 1405 LDA #$0E
.A 1407 STA $0800
.A 140A BRK
```

yazınız. (STA komutu, akümülatördeki değeri, belli bir bellek adresine yerleştir, anlamına gelir.)

Assembler'de yazdığınız programın makine dili karşılığı şöyledir:

A9 01
8D 00 04
A9 0E
8D 00 D8
00

Ekranı temizleyin ve;

.G 1400 yazın ve **RETURN** tuşuna basın.

Eğer her şeyi doğru yazdıysanız G harfinin değişik renkli bir A harfine dönüşmesi gerekir.

Artık ilk makine dili programınızı yazmış oldunuz. Programınızın amacı ekran belleğinin ilk yerleşimine "A" karakterini yerleştirmektir. Bunu başardıktan sonra şimdi diğer komutları ve kurallarını keşfetmeliyiz.

ADRESLEME ÇEŞİTLERİ

ZERO PAGE (0'ıncı sayfa)

Daha evvel gösterildiği gibi "absolute" (mutlak) adresler alt ve üst basamak olmak üzere iki baytla ifade edilirler. Üst basamak baytı, bellek sayfası olarak adlandırılır. Örneğin $1637 adresi, $16(22)'nci sayfada, $0277 adresi ise $02(2)'nci sayfadadır. ZERO PAGE denilen adresleme tipi, isminden de anlaşılacağı gibi 0'ıncı sayfadaki, yani, $0000 ile $00FF arasındaki bellek birimlerini adresi emekte kullanılır. Bu adreslerin üst basamak baytlar, daima 0'dır. Mutlak adres kullanımında 2 bayt kullanılmasına karşılık, 0'ıncı sayfa modunda, $0000 ile $00FF arasındaki bir adres, sadece bir baytla ifade edilebilir. Bu özellik şimdilik sizin için çok önemli görünmese de daha sonra kullanmanız gerekecektir.

STACK (YIĞIN ya da DESTE) ve Stack Pointer (Yığın Göstergeci)

Belleğin $0100 ile $01FF adresleri arasındaki 256 baytlık bölgesinin adıdır. Bu bölge, hem programcı hem de mikroişlemci tarafından program akışıyla ilgili bazı bilgileri kısa süreli olarak not etmekte kullanılır. En önemli kullanımı, BASIC veya makine dili bir program içinde alt-programlar (subroutine) işlendiği sırada, alt program bir RETURN veya RTS komutuyla sona erdiğinde geri dönülecek olan adresi saklamaktır.

Yığın, tıpkı bir oyun kâğıdı destesi gibi, son giren ilk çıkar (Last In First Out) ilkesine göre çalışır. Yığına atılan bir rakam $01FF'ten geriye doğru, yığının en üstüne girer: yığından bir rakam çekildiğinde en üstteki rakam çıkar.

Yığın göstergeci, yığındaki bir sonraki yazılabilir yeri gösterir. Yığına yazılan herhangi bir şey, yığın göstergecinin gösterdiği yere yazılır ve yazma işlemi bittikten sonra göstergeç azalarak bir alt adresi gösterir. Yığından bir bilgi alındığında ise göstergeç bir artar. Makine dilinde PHA komutu, akümülatörün o andaki değerini yığına atmakta, PLA komutu ise yığının üzerindeki bu değeri, akümülatöre aktarmakta kullanılır. PHP komutu statü kaydının o andaki değerini yığına atar: PLP ise destedeki son değeri statü kaydına aktarır.

İNDEKSLİ ADRESLEME

İndeksli adreslemede esas, istenen adresin, belli bir taban adresine X veya Y kayıtlarının o andaki değerinin eklenmesiyle bulunmasıdır. Örneğin X'te o anda $09 değeri varsa ve mikroişlemciye, akümülatöre $9000 taban adresinden X indeksli adresteki değerin yüklenmesi komutu verilmişse, akümülatöre 9000 + 09 = 9009 adresindeki değer yüklenecektir.

İndeksli bir komutu, mutlak veya sıfırıncı sayfa adresli bir komuttan ayıran tek fark, verilen adrese X veya Y ekleneceğini bildiren "X" veya "Y" ekidir.

Söylenenlerden de anlaşılabileceği gibi, indeksli adreslemenin dört tipi mümkündür: Mutlak, X indeksli: Mutlak, Y indeksli: sıfırıncı sayfa, X indeksli; sıfırıncı sayfa, Y indeksli.

DOLAYLI ADRESLEME

Dolaylı adreslemede, sıfırıncı sayfada tek baytla belirtilen bir adres verilir. Bu ve bunu izleyen adreste o anda bulunan değerler, mikroişlemci tarafından, asıl adres tabanı olarak kabul edilir.

DOLAYLI İNDEKSLİ ADRESLEME (INDIRECT INDEXED)

Bu adresleme tarzı, indeks olarak sadece Y kaydını kullanabilir. Sıfırıncı sayfada birbirini izleyen iki bayt halinde yazılmış olan adrese Y'nin değerini ekleyerek. Bulunan adresin işleneceğini belirtir.

Örneğin: $0020 adresinde 00, $0021 adresinde ise 90 değerinin bulunduğunu, Y'nin değerinin ise o anda 1F olduğunu varsayalım. Akümülatöre, $20 tabanında dolaylı ve Y indeksli değeri yüklemek LDA ($20), Y şeklinde yazılır ve şu anlama gelir:

$0020'deki değeri adres tabanının alt basamağı olarak al.
$0021'deki değeri adres tabanının üst basamağı olarak al.
Böylece bulduğun $9000 adresine Y'nin değerini ekle.
Böylece bulduğun $901F adresinde ne değer varsa akümülatöre yükle.

İNDEKSLİ DOLAYLI ADRESLEME (INDEXED INDIRECT)

Bu adresleme türü de indeks olarak sadece X'i kullanabilir. Sıfırıncı sayfada bir bayt olarak belirtilen adresten X ötedeki adreste ve onu izleyen adreste bulunan iki değerin adres olarak alınacağını gösterir. Örneğin $0030 adresinde 73, $0031 adresinde C0 ve X kaydında da 10 değerlerinin bulunduğunu düşünelim. Akümülatöre $20 tabanından X indeksli ve dolaylı değeri yüklemek. LDA ($20. X> şeklinde belirtilir ve şu anlama gelir:

$0020'ye X'in değerini ekle

Böylece bulduğun $0030 adresindeki değeri alt basamak kabul et $0031 adresindeki değeri üst basamak kabul et.

Böylece bulduğun $C073 adresindeki değeri akümülatöre yükle.

Buraya kadar sayılan adresleme çeşitlerini kısaca özetleyelim:

ADRESLEME TİPİ	MAKİNA DİLİ	ASSEMBLY	SONUÇ
1. Dolaysız (Immediate)	A9 20	LDA #$20	20
2. Mutlak (Absolute)	AD 20 00	LDA $0020	00
3. Sıfırıncı sayfa (Zero Page)	A5 20	LDA $20	00
4. Mutlak, Y indeksli	B9 20 00	LDA $0020,Y	90
5. Mutlak, X indeksli	BD 20 00	LDA $0020,X	73
6. Sıfırıncı sayfa, X indeksli	B5 20	LDA $20,X	73
7. Dolaylı indeksli	B1 20	LDA ($20),Y	13
8. İndeksli dolaylı	A1 20	LDA ($20,X)	A5

ANLAMLARI:

(Y=$01, X=$10, $0020=$00, $0021=$90, $0030=$73, $0031=$C0, $9001=$13, $C073 • $A5)

1. A'ya 20 Yükle
2. A'ya 0020'deki değeri yükle
3. A'ya 0020'deki değeri yükle
4. A'ya 0021'deki değeri yükle
5. A'ya 0030'daki değeri yükle
6. A'ya 0030'daki değeri yükle
7. A'ya 9001'deki değeri yükle
8. A'ya C073'teki değeri yükle

DALLANMA VE KOŞULLU KOMUTLAR

Makine dilinde bulunan bir diğer önemli olanak da CBM BASIC'de kullanılan "IF … THEN, IF, … GOTO" deyimleri ile yapılan koşullandırma ve belli durumların algılanması gibi işlemlerin yapılabilmesidir.

Statü kaydınsa bulunan bayraklar (bu, kayıttaki bitlere verilen değişik bir isimdir) çeşitli durumlardan değişik şekillerde etkilenir. Örneğin, herhangi bir komut sıfır sonucunun oluşmasına neden olursa, statü kaydındaki sıfır bayrağı (yani, birinci bit) 1 konumunu alır, sonuç sıfır olmadığı zaman ise sıfır bayrağı boş (0) durumundadır.

LDA # $00

komutu akümülatöre 0 değerini yüklediğinden sonuç 0 bayrağını etkiler, yani 1 yapar.

Belirli durumlar oluştuğunda programın bir başka kısma dallanması için kullanılan birtakım komutlar vardır. Dallanma komutlarına örnek olarak "sonuç sıfıra eşit olduğunda dallan" anlamına gelen BEQ komutunu verebiliriz. Bu komutlar eğer istenilen durum gerçekleştiyse dallanma olur, gerçekleşmediyse hiçbir işlem yapılmadan bir sonraki komuta geçilir. Dallanma komutları sadece bir evvelki komuta bakarak değil, durum kaydını inceleyerek de çalışabilir. Biraz evvel bahsedildiği gibi durum kaydının içinde bir "sonuç sıfır" bayrağı vardır. BEQ komutu bu bayrak (Z olarak bilinir) 1 olursa çalışır. Bu komuta zıt olarak çalışan BNE komutu ise sonucun sıfır olmadığı zamanlarda dallanma olmasına sebep olur.

X ve Y kayıtları, o andaki değerlerini bir arttıran veya bir eksilten birtakım komutlara sahiptir. Örneğin. INX komutu X indeksini bir arttırır. X, arttırılmadan önce $FF değerine (sahip olabileceği en büyük değer) sahipse, bu komutla tekrar sıfır değerine döner. Eğer programınızın, X'in değeri sıfırlanıncaya kadar bir işlemi tekrarlamasını istiyorsanız. BNE komutunu X'teki değer için kullanıp sürekli döngü yapabilirsiniz.

INX'in tersi, X'i bir azaltan DEX komutudur. Değer sıfır ise, DEX komutu ile bu değer $FF olur. Y için de aynı işlemler INY ve DEY komutları ile yaptırılır.

Eğer yazacağınız bir programda X ve Y, o değeri almadan dallanma olmasını istiyorsanız ne yapmalısınız? Böyle problemler için de CPX, CPY gibi karşılaştırma komutları vardır.

Bu komutlar ile makine dili programcısı, indeks kayıtlarını belirli dolaysız değerlerle veya belleğin bir adresindeki değerle karşılaştırılabilir. Örneğin X kaydında $40 değerinin olup olmadığını öğrenmek istiyorsanız, şu komutu kullanacaksınız:

```
CPX # $40    - X'i $40 değeri ile karşılaştır.
BEQ          - Eğer durum sağlanırsa programın belirtilen noktasına geç
```

Makine dili programlarda, dallanma ve karşılaştırma komutları çok önemli bir yer tutar.

64MON kullanırken dallanma komutlarında belirtilen işlenen (operand) değerleri, istenilen durumlar oluştuğunda, programın dallanacağı adrestir. Yani, programın o an bulunduğu yerden, burada verilen uzaklıktaki adrese gitmesi sağlanır. Dallanılan uzaklık, sadece bir baytlık olabilir. Dallanma 127 bayt ileriye, 128 bayt da geriye doğru yapılabilir.

> **NOT:** Bu 255 baytın toplam aralığıdır. Çünkü bir baytın içerebileceği en büyük değer, 255'tir

Bu sınırı aştığımızda 64MON, komutu çevirmeyi reddeder. Dallanma komutlarını şimdilik çok fazla kullanmayacağımızdan, bu problemi fazla önemsemeyin.

Uzaklığı verilen adrese dallanma yöntemi, mutlak adresleme moduna uymaz. Dallanma komutları makine dili standartlarına göre hızlı komutlardır. 64MON sizin mutlak adresleme modunu kullanmanıza izin verir, daha sonra kendisi doğru uzaklığı hesaplar. Bu, çevirici kullanmanın sağladığı yararlardan biridir. (Örneğin; $C000 adresine BNE $C004 yazmak istediğinizde, çevirici $C000-$C004 arasındaki uzaklığı kendiliğinden hesaplayarak D0 02 kodunu oluşturacaktır).

ALT PROGRAMLAR:

Makine dilini kullanırken (BASIC dilinde olduğu gibi), alt programları çağırabilirsiniz. Çağırma işlemi için, JSA komutu ve bir mutlak adres gereklidir.

İşletim sistemiyle beraber olarak, ekrana karakter yazan, makine dili bir alt program vardır. Bu altprogram çağırılmadan önce, karakterin CBM ASCII kodu akümülatöre yazılmalıdır. Alt programın adresi $FFD2'dir.

Buna göre ekrana “AD” yazabilmek için, aşağıdaki programın yazılması gerekir.

```
.A 1400 LDA #$41
.A 1402 JSR $FFD2
.A 1405 LDA #$44
.A 1407 JSR $FFD2
.A 140A LDA #$0D
.A 140C JSR $FFD2
.A 140F BRK
.G 1400
```

1400 -akümülatöre "A" 'nın CBM ASCIl kodunu yükle
1402 -ekrana yaz
1405 -akümülatöre "D" 'nin CBM ASCII kodunu yükle
1407 -ekrana yaz
140A -satırbaşı işaretini yükle (yani [RETURN])
140F -programı sonlardır 64MON'a dön
Program ekrana "AD" basar ve 64MON'a geri döner.

Yukarıda, ekrana bir karakter yazdırmayı sağlayan kısa program için, KERNAL’ın sıçrama tablosunun bir kısmını kullandık. BASIC'de kullanılan GOTO komutuyla aynı işlemi gören JMP Komutu, belirli bir adrese atlanılması için kullanılır. KERNAL Commodore 64’ün bütün giriş çıkışlarını kontrol eden standartlaştırılmış altprogramların uzun bir listesini içerir. KERNAL JMP'ların her biri, işletim sisteminde bir altprograma karşılık gelir. Bu sıçrama tablosu işletim sisteminde $FF84 ve $FFF5 arasındaki bellek yerleşimlerini kaplar. KERNAL'ın tam açıklaması kitabın daha sonraki bölümlerinde yer alıyor.

Şimdi öğrendiğiniz bu yeni prensipleri başka bir programda kullanalım.

Programın amacı kısa bir KERNAL programı ile alfabeyi ekrana yazmaktır. Kullanılan tek yeni komut, X'in içindekileri akümülatöre transfer eden TXA komutudur.

```
.A 1400 LDX #$41
.A 1402 TXA
.A 1403 JSR $FFD2
.A 1406 INX
.A 1407 CPX #$5B
.A 1409 BNE $1402
.A 140B BRK
```

Commodore 64'ün alfabeyi yazışını görmek istiyorsanız, bildiğiniz komutu yazın.

.G 1400

Programın yanına yazılan yorumlar program akışını ve mantığını açıklamak için kullanılmıştır. Eğer siz bir program yazarsanız önce kâğıda yazın sonra da mümkünse küçük parçalar halinde deneyin.

YENİ BAŞLAYANLAR İÇİN YARARLI İPUÇLARI

Makine dilini öğrenmek için en kolay yol başkalarının programlarını incelemektir. 6510 (veya 6502) mikroişlemcisi kullanan başka bir bilgisayar için yazılmış olsa bile incelemeniz yararlı olacaktır. Baktığınız zaman kodu tamamen anlayabildiğinizden emin olmalısınız. Bu iş biraz sabır ister. İlk zamanlar sizi çok öfkelendirirse de sabrınız devam ettiği sürece galibin siz olacağını göreceksiniz.

Programların makine dili ile yazılmasının avantajları:

1. Hız - Makine dili, BASIC gibi yüksek seviyeli programlama dilinden yüz, hatta bazı özel durumlarda bin defa daha hızlı çalışabilir.

2. Sıkılık - Makine dili programlar "su sızmaz" sıkılıkta yapılabilir. Böylece kullanıcının, programın izin verdiğinden fazlasını yapması önlenir. Yüksek seviyeli bir dille uğraşan bir kullanıcının, örneğin bir sıfır girerek, BASIC yorumlayıcısı tarafından verilecek:

?DIVISION BY ZERO ERROR IN LINE 830 (830 numaralı satırda sıfıra bölünme hatası) hatasına neden olmayacağından emin olabilirsiniz.

Kısaca, bilgisayardan en iyi şekilde makine dili programcısı faydalanabilir.

BÜYÜK BİR İŞE BAŞLARKEN

Makine diliyle büyük bir işe kalkışırken, genellikle kafalarda bazı şüpheler oluşur. Bazı işlemlerin nasıl yaptırılacağı düşünülür. Programı ilk önce kâğıda yazınız. Yazarken de bellek kullanımı ile ilgili blok diyagramları, gereken kodların fonksiyonel birimlerini ve program akış şemasını kullanınız. Makine diliyle rulet oyunu yazmak istediğinizi düşünelim. Özeti şöyle olacaktır:

- Görüntü başlığı
- Oyuncunun bilgi isteyip istemediğinin sorulması
- EVET - listenin gösterilmesi – BAŞLANGIÇ'A gidilmesi
- HAYIR - BAŞLANGIÇ'A gidilmesi
- BAŞLANGIÇ - her şeyin ilk başlama durumuna getirilmesi
- ANA - rulet masasının gösterimi
- Bahislerin alınması
- Tekerleğin dönmesi
- Durması için tekerleğin yavaşlaması
- Bahislerin sonuçlarla karşılaştırılması
- Oyuncunun sonuçtan haberdar edilmesi
- Oyuncunun parası kaldı mı?
- EVET - ANA bölüme git
- HAYIR - Kullanıcıyı haberdar et! BAŞLANGIÇ'A git

Özeti verilen bu programda her birime gelindiğinde birim kendi içinde parçalara ayrılarak programın daha kolay bir biçimde yapılması sağlanabilir.

Bu, program pratiğiniz için oldukça faydalı bir programdır. UĞRAŞIN!

6510 MİKRO İŞLEMCİSİ KOMUT KÜMESİ – ALFABETİK SIRA İLE

ADC	Belleği taşıma ile beraber akümülatöre ekle
AND	Belleği akümülatör ile “AND” (VE) işlemine sok
ASL	Bir bit sola kaydır (bellek veya akümülatörü)
BCC	Taşıma sıfır ise dallan
BCS	Taşıma sıfır değilse dallan
BEQ	Sonuç sıfır ise dallan
BIT	Bellekteki bitleri akümülatördekilerle karşılaştır
BMI	Sonuç negatif ise dallan
BME	Sonuç sıfır değil ise dallan
BPL	Sonuç pozitif ise dallan
BRK	Bitir
BVC	Overflow (taşma) biti sıfır ise dallan
BVS	Overflow (taşma) biti bir ise dallan
CLC	Elde bayrağını 0’la
CLD	Ondalık moddan çık
CLI	Kesinti önleme bitini sıfırla
CLV	Taşma bayrağını sıfırla
CMP	Bellekle akümülatörü karşılaştır
CPX	Bellekle indeks kaydı X’i karşılaştır
CPY	Bellekle indeks kaydı Y’i karşılaştır
DEC	Bellekteki değeri bir azalt
DEX	X indeksini bir azalt
DEV	Y indeksini bir azalt
EOR	Belleği akümülatörle “Exclusive – Or” işlemine sok
INC	Bellekteki değeri bir arttır.
INX	X indeksini bir arttır
INY	Y indeksini bir arttır
JMP	Belirtilen adresi atla
JSR	Dönüş adresini saklayarak belirtilen adrese atla
LDA	Bellekteki değeri akümülatöre yükle
LDX	Bellekteki değeri X indeksine yükle
LDY	Bellekteki değeri Y indeksine yükle
LSR	Bir bit sağa kaydır (bellek veya akümülatörü)
NOP	Hiçbir işlem yapma
ORA	Belleği akümülatör ile “OR” (VEYA) işlemine sok
PHA	Akümülatörü yığına (Stack) gönder
PHP	İşlemcinin durum değerini yığına gönder
PLA	Akümülatörü yığından al
PLP	İşlemcinin durum değerini yığından al
ROL	Bir bit sola döndür (bellek veya akümülatör)
ROR	Bir bit sağa döndür (bellek veya akümülatörü)
RTI	Kesinti işleminden geri dön
RTS	Altprogramdan dön
SBC	Ödünç ile birlikte belleği akümülatörden çıkart
SEC	Kalan bayrağını 1 yap
SED	Ondalık moduna gir
SEI	Kesinti önleme durum bitini bir yap
STA	Akümülatördeki değeri belleğe sakla
STX	X indeksini belleğe sakla
STY	Y indeksini belleğe sakla
TAX	Akümülatörün değerini X indeksine gönder
TAY	Akümülatörün değerini Y indeksine gönder
TSX	Yığın göstergecini X indeksine gönder
TXA	X indeksini akümülatöre gönder
TXS	X indeksini yığın göstergecine gönder
TYA	Y indeksini akümülatöre gönder

Aşağıdakiler, kitabın geri kalan kısımlarında kısaltılmış olarak kullanılacaklardır;

A	Akümülatör
X,Y	İndeks kayıtları
M	Bellek (Memory)
P	İşlemci durum kaydı (Processor Status Register)
S	Yığın göstergeci (Stack Pointer)
√	Değişikliğe uğrar
_	Değişme yok
+	Topla
Λ	Mantıksal VE (Logical AND)
-	Çıkarma
V̶	Mantıksal özel VE "Exlusive Or"
↑	Yığından transfer
↓	Yığına transfer
→	Birime transfer
←	Birimden transfer
V	Mantıksal VEYA (Logical OR)
PC	Program sayacı (Program Counter)
PCH	Program sayacı üst (Program Counter High)
PCL	Program sayacı alt (Program Counter Low)
OPER	İşlenen (OPERAND)
#	Dolaysız adres modu (IMMEDIATE ADDRESSING MODE)

NOT: Aşağıda gösterilecek her tablonun başında, komutun MCS6500 Microcomputer Family Programming Manual isimli kitaptaki yerleri belirtilmiştir. (Örneğin: Ref:2.2.30 gibi.)
Ayrıca, "çevrim sayısı" ile de C64 ana işlem ünitesinin söz konusu komut için ne kadar süre harcadığı belirtilmektedir. C64, saniyede 1 milyon 200 bin civarında çevrim (cycle) hızına sahiptir. Yani 1.2 MHz hızla çalışır.

ADC

Add memory to accumulator with carry
Belleği taşıma ile beraber akümülatöre ekle
İşlem: **A + M + C → A, C**

N	Z	C	I	D	V
√	√	√	_	_	√

(Ref : 2.2.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	ADC # Oper	69	2	2
Zero Page	ADC Oper	65	2	3
Zero Page, X	ADC Oper, X	75	2	4
Absolute	ADC Oper	6D	3	4
Absolute, X	ADC Oper, X	7D	3	4*
Absolute, Y	ADC Oper, Y	79	3	4*
(Indirect, X)	ADC (Oper, X)	61	2	6
(Indirect), Y	ADC (Oper), Y	71	2	5*

*Sayfa sınırı aşılırsa 1 ekleyin.

AND

"AND" memory with accumulator
Belleği akümülatörle "VE" işlemine sok
İşlem: **A ∧ M → A**

N Z C I D V
√ √ _ _ _ _

(Ref : 2.2.3.0)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	AND # Oper	29	2	2
Zero Page	AND Oper	25	2	3
Zero Page, X	AND Oper, X	35	2	4
Absolute	AND Oper	2D	3	4
Absolute, X	AND Oper, X	3D	3	4*
Absolute, Y	AND Oper, Y	39	3	4*
(Indirect, X)	AND (Oper, X)	21	2	6
(Indirect), Y	AND (Oper), Y	31	2	5*

*Sayfa sınırı aşılırsa 1 ekleyin.

ASL

Shift left one bit (memory or accumulator)
Bir bit sola kaydır (bellek veya akümülatörü)
İşlem: **C ← |7|6|5|4|3|2|1|0| ← 0**

N Z C I D V
√ √ √ _ _ _

(Ref : 10.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Accumulator	ASL A	0A	1	2
Zero Page	ASL Oper	06	2	5
Zero Page, X	ASL Oper, X	16	2	6
Absolute	ASL Oper	0E	3	6
Absolute, X	ASL Oper, X	1E	3	7

BBC

Branch on carry clear
Taşıma biti 0 ise dallan
İşlem: **C = 0 ise dallan**

N Z C I D V
_ _ _ _ _ _

(Ref : 4.1.1.3)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Relative	BBC Oper	90	2	2*

*Aynı sayfada dallanma meydana gelirse 1 ekleyin.
*Farklı bir sayfaya dallanma meydana gelirse 2 ekleyin.

BCS Branch on carry set

Taşıma biti 1 ise dallan

İşlem: **C = 1 ise dallan**

N Ƶ C I D V

– – – – – –

(Ref : 4.1.1.4)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Relative	BCS Oper	B0	2	2*

*Aynı sayfada dallanma meydana gelirse 1 ekleyin.
*Farklı bir sayfaya dallanma meydana gelirse 2 ekleyin.

BEQ Branch on result zero

Sonuç 0 ise dallan

İşlem: **Ƶ = 1 ise dallan**

N Ƶ C I D V

– – – – – –

(Ref : 4.1.1.5)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Relative	BEQ Oper	F0	2	2*

*Aynı sayfada dallanma meydana gelirse 1 ekleyin.
*Sonraki sayfada dallanma meydana gelirse 2 ekleyin.

BIT Test bits in memory with accumulator

Bellekteki bitleri akümülatördekilerle karşılaştırır

İşlem: **A Λ M, M7 → N, M6 → V**

N Ƶ C I D V

M_7 √ – – – M_6

-6 ve 7'nci bitler durum kaydına yerleştirilir.
-Eğer A /\ M'in sonucu 0 ise Ƶ = 1'dir. Değil ise Ƶ = 0'dır.

(Ref : 4.2.1.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Zero Page	BIT Oper	24	2	3
Absolute	BIT Oper	2C	3	4

BMI Branch on result minus

Sonuç eksi ise dallan

İşlem: **N = 1 ise dallan**

N Ƶ C I D V

– – – – – –

(Ref : 4.1.1.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Relative	BMI Oper	30	2	2*

*Aynı sayfada dallanma meydana gelirse 1 ekleyin.
*Farklı bir sayfaya dallanma meydana gelirse 2 ekleyin.

BNE Branch on result not zero

Sonuç sıfır değil ise dallan

İşlem: **Ƶ = 0 ise dallan**

N	Ƶ	C	I	D	V
–	–	–	–	–	–

(Ref : 4.1.1.6)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Relative	BNE Oper	D0	2	2*

*Aynı sayfada dallanma meydana gelirse 1 ekleyin.
*Farklı bir sayfaya dallanma meydana gelirse 2 ekleyin.

BPL Branch on result plus

Sonuç pozitif ise dallan

İşlem: **N = 0 ise dallan**

N	Ƶ	C	I	D	V
–	–	–	–	–	–

(Ref : 4.1.1.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Relative	BPL Oper	10	2	2*

*Aynı sayfada dallanma meydana gelirse 1 ekleyin.
*Farklı bir sayfaya dallanma meydana gelirse 2 ekleyin.

BRK Force break

Bırak

İşlem: **PC + 2 ↓ P ↓ programı keser**

N	Ƶ	C	I	D	V
–	–	–	1	–	–

(Ref : 9.11)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	BRK	00	1	7

1. Bir BRK komutu I ayarıyla maskelenemez.

BVC Branch on overflow clear

Taşma biti sıfır ise dallan

İşlem: **V = 0 ise dallan**

N	Ƶ	C	I	D	V
–	–	–	–	–	–

(Ref : 4.1.1.8)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Relative	BVC Oper	50	2	2*

*Aynı sayfada dallanma meydana gelirse 1 ekleyin.
*Farklı bir sayfaya dallanma meydana gelirse 2 ekleyin.

BVS Branch on overflow set

Taşma biti bir ise dallan

İşlem: **V = 1 ise dallan**

N	Ƶ	C	I	D	V
_	_	_	_	_	_

(Ref : 4.1.1.7)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Relative	BVS Oper	70	2	2*

*Aynı sayfada dallanma meydana gelirse 1 ekleyin.

*Farklı bir sayfaya dallanma meydana gelirse 2 ekleyin.

CLC Clear carry flag

Taşıma bayrağını sıfırla

İşlem: **0 → C**

N	Ƶ	C	I	D	V
_	_	0	_	_	_

(Ref : 3.0.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	CLC	18	1	2

CLD Clear decimal mode

Ondalık moddan çık

İşlem: **0 → D**

N	Ƶ	C	I	D	V
_	_	_	_	0	_

(Ref : 3.3.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	CLD	D8	1	2

CLI Clear interrupt disable bit

Kesinti önleme bitini sıfırla

İşlem: **0 → I**

N	Ƶ	C	I	D	V
_	_	_	0	_	_

(Ref : 3.2.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	CLI	58	1	2

CLV Clear overflow flag

Taşma bayrağını sıfırla

İşlem: **0 → V**

N	Z	C	I	D	V
–	–	–	–	–	0

(Ref : 3.6.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	CLV	B8	1	2

CMP Compare memory and accumulator

Bellekle ile akümülatörü karşılaştır

İşlem: **A – M**

N	Z	C	I	D	V
√	√	√	–	–	–

(Ref : 4.2.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	CMP # Oper	C9	2	2
Zero Page	CMP Oper	C5	2	3
Zero Page, X	CMP Oper, X	D5	2	4
Absolute	CMP Oper	CD	3	4
Absolute, X	CMP Oper, X	DD	3	4*
Absolute, Y	CMP Oper, Y	D9	3	4*
(Indirect, X)	CMP (Oper, X)	C1	2	6
(Indirect), Y	CMP (Oper), Y	D1	2	5*

*Sayfa sınırı aşılırsa 1 ekleyin.

CPX Compare memory and index X

Bellekle indeks kaydı X'i karşılaştır

İşlem: **X – M**

N	Z	C	I	D	V
√	√	√	–	–	–

(Ref : 7.8)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	CPX # Oper	E0	2	2
Zero Page	CPX Oper	E4	2	3
Absolute	CPX Oper	EC	3	4

CPY Compare memory and index Y
Bellekle indeks kaydı Y'i karşılaştır
İşlem: **Y – M**

N Z C I D V
√ √ √ _ _ _

(Ref : 7.9)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	CPY # Oper	C0	2	2
Zero Page	CPY Oper	C4	2	3
Absolute	CPY Oper	CC	3	4

DEC Decrement memory by one
Bellekteki değeri 1 azalt
İşlem: **M – 1 → M**

N Z C I D V
√ √ _ _ _ _

(Ref : 10.7)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Zero Page	DEC Oper	C6	2	5
Zero Page, X	DEC Oper, X	D6	2	6
Absolute	DEC Oper	CE	3	6
Absolute, X	DEC Oper, X	DE	3	7

DEX Decrement index X by one
X indeksini 1 azalt
İşlem: **X – 1 → X**

N Z C I D V
√ √ _ _ _ _

(Ref : 7.6)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	DEX	CA	1	2

DEY Decrement index Y by one
Y indeksini 1 azalt
İşlem: **Y – 1 → Y**

N Z C I D V
√ √ _ _ _ _

(Ref : 7.7)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	DEY	88	1	2

EOR "Exclusive-OR" memory with accumulator
Belleği akümülatörle "Özel-VEYA" işlemine sok
İşlem: $A \veebar M \rightarrow A$

N	Ƶ	C	I	D	V
√	√	–	–	–	–

(Ref : 2.2.3.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	EOR # Oper	49	2	2
Zero Page	EOR Oper	45	2	3
Zero Page, X	EOR Oper, X	55	2	4
Absolute	EOR Oper	4D	3	4
Absolute, X	EOR Oper, X	5D	3	4*
Absolute, Y	EOR Oper, Y	59	3	4*
(Indirect, X)	EOR (Oper, X)	41	2	6
(Indirect), Y	EOR (Oper), Y	51	2	5*

*Sayfa sınırı aşılırsa 1 ekleyin.

INC Increment memory by one
Bellekteki değeri 1 arttır
İşlem: $M + 1 \rightarrow M$

N	Ƶ	C	I	D	V
√	√	–	–	–	–

(Ref : 10.6)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Zero Page	INC Oper	E6	2	5
Zero Page, X	INC Oper, X	F6	2	6
Absolute	INC Oper	EE	3	6
Absolute, X	INC Oper, X	FE	3	7

INX Increment index X by one
X indeksini 1 arttır
İşlem: $X + 1 \rightarrow X$

N	Ƶ	C	I	D	V
√	√	–	–	–	–

(Ref : 7.4)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	INX	E8	1	2

INY Increment index Y by one
Y indeksini 1 arttır
İşlem: $Y + 1 \rightarrow Y$

N	Ƶ	C	I	D	V
√	√	–	–	–	–

(Ref : 7.5)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	INY	C8	1	2

JMP Jump to new location

Yeni konuma atla

İşlem: **(PC + 1) → PCL**

(PC + 2) → PCH

N	Z	C	I	D	V
_	_	_	_	_	_

(Ref : 4.0.2)

(Ref : 9.8.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Absolute	JMP Oper	4C	3	3
Indirect	JMP (Oper)	6C	3	5

JSR Jump to new location saving return address

Dönüş adresini saklayarak yeni konuma atla

İşlem: **PC + 2 ↓, (PC + 1) → PCL**

(PC + 2) → PCH

N	Z	C	I	D	V
_	_	_	_	_	_

(Ref : 8.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Absolute	JSR Oper	20	3	6

LDA Load accumulator with memory

Bellekteki değeri akümülatöre yükle

İşlem: **M → A**

N	Z	C	I	D	V
√	√	_	_	_	_

(Ref : 2.1.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	LDA # Oper	A9	2	2
Zero Page	LDA Oper	A5	2	3
Zero Page, X	LDA Oper, X	B5	2	4
Absolute	LDA Oper	AD	3	4
Absolute, X	LDA Oper, X	BD	3	4*
Absolute, Y	LDA Oper, Y	B9	3	4*
(Indirect, X)	LDA (Oper, X)	A1	2	6
(Indirect), Y	LDA (Oper), Y	B1	2	5*

* Sayfa sınırı aşılırsa 1 ekleyin.

LDX Load index X with memory
Bellekteki değeri X indeksine yükle
İşlem: **M → X**

N	Z	C	I	D	V
√	√	_	_	_	_

(Ref : 7.0)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	LDX # Oper	A2	2	2
Zero Page	LDX Oper	A6	2	3
Zero Page, Y	LDX Oper, Y	B6	2	4
Absolute	LDX Oper	AE	3	4
Absolute, Y	LDX Oper, Y	BE	3	4*

* Sayfa sınırı aşıldığında 1 ekleyin.

LDY Load index Y with memory
Bellekteki değeri Y indeksine yükle
İşlem: **M → Y**

N	Z	C	I	D	V
√	√	_	_	_	_

(Ref : 7.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	LDY # Oper	A0	2	2
Zero Page	LDY Oper	A4	2	3
Zero Page, X	LDY Oper, X	B4	2	4
Absolute	LDY Oper	AC	3	4
Absolute, X	LDY Oper, X	BC	3	4*

* Sayfa sınırı aşıldığında 1 ekleyin.

LSR Shift right one bit (memory or accumulator)
Bir bit sağa kaydır (bellek veya akümülatörü)
İşlem: **0 → [7|6|5|4|3|2|1|0] → C**

N	Z	C	I	D	V
0	√	√	_	_	_

(Ref : 10.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Accumulator	LSR A	4A	1	2
Zero Page	LSR Oper	46	2	5
Zero Page, X	LSR Oper, X	56	2	6
Absolute	LSR Oper	4E	3	6
Absolute, X	LSR Oper, X	5E	3	7

NOP No operation
İşlem yok
İşlem: **İşlem yok (2 döngü)**

N Z C I D V
– – – – – –

(Ref :)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	NOP	EA	1	2

ORA "OR" memory with accumulator
Belleği akümülatör ile "VEYA" işlemine sok
İşlem: **A V M → A**

N Z C I D V
√ √ – – – –

(Ref : 2.2.3.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	ORA # Oper	09	2	2
Zero Page	ORA Oper	05	2	3
Zero Page, X	ORA Oper, X	15	2	4
Absolute	ORA Oper	0D	3	4
Absolute, X	ORA Oper, X	1D	3	4*
Absolute, Y	ORA Oper, Y	19	3	4*
(Indirect, X)	ORA (Oper, X)	01	2	6
(Indirect), Y	ORA (Oper), Y	11	2	5

* Sayfa geçişine 1 ekleyin.

PHA Push accumulator on stack
Akümülatörü yığına gönder
İşlem: **A ↓**

N Z C I D V
– – – – – –

(Ref : 8.5)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	PHA	48	1	3

PHP Push processor status on stack
İşlemcinin durumunu yığına gönder
İşlem: **P ↓**

N Z C I D V
– – – – – –

(Ref : 8.11)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	PHP	08	1	3

PLA Pull accumulator from stack

Akümülatörü yığından al

İşlem: **A ↑**

N Ƶ C I D V
√ √ _ _ _ _

(Ref : 8.6)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	PLA	68	1	4

PLP Pull processor status from stack

İşlemcinin durum değerini yığından al

İşlem: **P ↑**

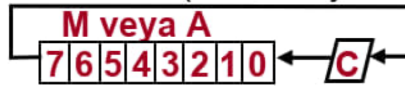
N Ƶ C I D V
YIĞINDAN

(Ref : 8.12)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	PLP	28	1	4

ROL Rotate one bit left (memory or accumulator)

Bir bit sola döndür (bellek veya akümülatörü)

İşlem: 

N Ƶ C I D V
√ √ √ _ _ _

(Ref : 10.3)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Accumulator	ROL A	2A	1	2
Zero Page	ROL Oper	26	2	5
Zero Page, X	ROL Oper, X	36	2	6
Absolute	ROL Oper	2E	3	6
Absolute, X	ROL Oper, X	3E	3	7

ROR Rotate one bit right (memory or accumulator)

Bir bit sağa döndür (bellek veya akümülatörü)

İşlem: 

N Ƶ C I D V
√ √ √ _ _ _

(Ref : 10.4)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Accumulator	ROR A	6A	1	2
Zero Page	ROR Oper	66	2	5
Zero Page, X	ROR Oper, X	76	2	6
Absolute	ROR Oper	6E	3	6
Absolute, X	ROR Oper, X	7E	3	7

ROR komutu MCS650X mikroişlemcilerde Haziran 1976'dan sonra mevcut oldu.

RTI

Return from interrupt
Kesinti işleminden geri dön
İşlem: **P ↑ PC ↑**

N Z C I D V
YIĞINDAN

(Ref : 9.6)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	RTI	40	1	6

RTS

Return from subroutine
Alt programdan geri dön
İşlem: **PC ↑, PC + 1 → PC**

N Z C I D V
_ _ _ _ _ _

(Ref : 8.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	RTS	60	1	6

SBC

Subtract memory from accumulator with borrow
Ödünç alarak belleği akümülatörden çıkart
İşlem: **A – M – $\overline{C}$ → A**
Not: $\overline{C}$ = Ödünç almak (Carry'nin tersi)

N Z C I D V
√ √ √ _ _ √

(Ref : 2.2.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Immediate	SBC # Oper	E9	2	2
Zero Page	SBC Oper	E5	2	3
Zero Page, X	SBC Oper, X	F5	2	4
Absolute	SBC Oper	ED	3	4
Absolute, X	SBC Oper, X	FD	3	4*
Absolute, Y	SBC Oper, Y	F9	3	4*
(Indirect, X)	SBC (Oper, X)	E1	2	6
(Indirect), Y	SBC (Oper), Y	F1	2	5*

*Sayfa sınırı aşıldığında 1 ekleyin.

SEC

Set carry flag
Taşıma bayrağını 1 yap
İşlem: **1 → C**

N Z C I D V
_ _ 1 _ _ _

(Ref : 3.0.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	SEC	38	1	2

SED Set decimal mode
Ondalık modu ayarla
İşlem: **1 → D**

N	Z	C	I	D	V
_	_	_	_	1	_

(Ref : 3.3.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	SED	F8	1	2

SEI Set interrupt disable status
Kesintiyi devre dışı bırakma durumunu ayarla
İşlem: **1 → I**

N	Z	C	I	D	V
_	_	_	1	_	_

(Ref : 3.2.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	SEI	78	1	2

STA Store accumulator in memory
Akümülatörü bellekte sakla
İşlem: **A → M**

N	Z	C	I	D	V
_	_	_	_	_	_

(Ref : 2.1.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Zero Page	STA Oper	85	2	3
Zero Page, X	STA Oper, X	95	2	4
Absolute	STA Oper	8D	3	4
Absolute, X	STA Oper, X	9D	3	5
Absolute, Y	STA Oper, Y	99	3	5
(Indirect, X)	STA (Oper, X)	81	2	6
(Indirect), Y	STA (Oper), Y	91	2	6

STX Store index X in memory
X indeksini bellekte sakla
İşlem: **X → M**

N	Z	C	I	D	V
_	_	_	_	_	_

(Ref : 7.2)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Zero Page	STX Oper	86	2	3
Zero Page, Y	STX Oper, Y	96	2	4
Absolute	STX Oper	8E	3	4

STY Store index Y in memory
Y indeksini bellekte sakla
İşlem: **Y → M**

N Z C I D V
– – – – – –

(Ref : 7.3)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Zero Page	STY Oper	84	2	3
Zero Page, X	STY Oper, X	94	2	4
Absolute	STY Oper	8C	3	4

TAX Transfer accumulator to index X
Akümülatörü X indeksine aktar
İşlem: **A → X**

N Z C I D V
√ √ – – – –

(Ref : 7.11)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	TAX	AA	1	2

TAY Transfer accumulator to index Y
Akümülatörü Y indeksine aktar
İşlem: **A → Y**

N Z C I D V
√ √ – – – –

(Ref : 7.13)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	TAY	A8	1	2

TSX Transfer stack pointer to index X
Yığın işaretçisini X indeksine aktar
İşlem: **S → X**

N Z C I D V
√ √ – – – –

(Ref : 2.2.1)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	TSX	BA	1	2

TXA Transfer index X to accumulator
X indeksini akümülatöre aktar
İşlem: **X → A**

N Z C I D V
√ √ _ _ _ _

(Ref : 7.12)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	TXA	8A	1	2

TXS Transfer index X to stack pointer
X indeksini yığın işaretçisine aktar
İşlem: **X → S**

N Z C I D V
√ √ _ _ _ _

(Ref : 8.8)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	TXS	9A	1	2

TYA Transfer index Y to accumulator
Y indeksini akümülatöre aktar
İşlem: **Y → A**

N Z C I D V
√ √ _ _ _ _

(Ref : 7.14)

Adresleme Modu	Assembly Dili Biçimi	OP Kodu	Bayt Sayısı	Çevrim Sayısı
Implied	TYA	98	1	2

ADRESLEME MODLARI VE İŞLETİM ZAMANLARI (clock cycle: saat çevrimi)

	Acculumator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ADC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
AND	.	2	2	2	.	4	4*	4*	.	.	6	5*	.
ASL	2	.	5	6	.	6	7	.	.	.	.	.	.
BCC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BCS	.	.	.	.	.	.	.	.	.	2**	.	.	.
BEQ	.	.	.	.	.	.	.	.	.	2**	.	.	.
BIT	.	.	3	.	.	4	.	.	.	.	.	.	.
BMI	.	.	.	.	.	.	.	.	.	2**	.	.	.
BME	.	.	.	.	.	.	.	.	.	2**	.	.	.
BPL	.	.	.	.	.	.	.	.	.	2**	.	.	.
BRK	.	.	.	.	.	.	.	.	.	.	.	.	.
BVC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BVS	.	.	.	.	.	.	.	.	.	2**	.	.	.
CLC	.	.	.	.	.	.	.	.	2	.	.	.	.
CLD	.	.	.	.	.	.	.	.	2	.	.	.	.
CLI	.	.	.	.	.	.	.	.	2	.	.	.	.
CLV	.	.	.	.	.	.	.	.	2	.	.	.	.
CMP	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
CPX	.	2	3	.	.	4	.	.	.	.	.	.	.
CPY	.	2	3	.	.	4	.	.	.	.	.	.	.
DEC	.	.	5	6	.	6	7	.	.	.	.	.	.
DEX	.	.	.	.	.	.	.	.	2	.	.	.	.
DEV	.	.	.	.	.	.	.	.	2	.	.	.	.
EOR	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
INC	.	.	5	6	.	6	7	.	.	.	.	.	.
INX	.	.	.	.	.	.	.	.	2	.	.	.	.
INY	.	.	.	.	.	.	.	.	2	.	.	.	.
JMP	.	.	.	.	.	3	.	.	.	.	.	.	5
JSR	.	.	.	.	.	6	.	.	.	.	.	.	.
LDA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
LDX	.	2	3	.	4	4	.	4*	.	.	.	.	.
LDY	.	2	3	4	.	4	4*	.	.	.	.	.	.
LSR	2	.	5	6	.	6	7	.	.	.	.	.	.
NOP	.	.	.	.	.	.	.	.	2	.	.	.	.

* Sayfa sınırı boyunca indeksleniyorsa bir döngü ekleyin.
**Dallanma alınmışsa bir döngü ekleyin. Dallanma işlemi sayfa sınırını aşarsa bir döngü daha ekleyin.

ADRESLEME MODLARI VE İŞLETİM ZAMANLARI (clock cycle: saat çevrimi)

	Acculumator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ORA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
PHA	.	.	.	.	.	.	.	.	3	.	.	.	.
PHP	.	.	.	.	.	.	.	.	3	.	.	.	.
PLA	.	.	.	.	.	.	.	.	4	.	.	.	.
PLP	.	.	.	.	.	.	.	.	4	.	.	.	.
ROL	2	.	5	6	.	6	7	.	.	.	.	.	.
ROR	2	.	5	6	.	6	7	.	.	.	.	.	.
RTI	.	.	.	.	.	.	.	.	6	.	.	.	.
RTS	.	.	.	.	.	.	.	.	6	.	.	.	.
SBC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
SEC	.	.	.	.	.	.	.	.	2	.	.	.	.
SED	.	.	.	.	.	.	.	.	2	.	.	.	.
SEI	.	.	.	.	.	.	.	.	2	.	.	.	.
STA	.	.	3	4	.	4	5	5	.	.	6	6	.
STX	.	.	3	.	4	4	.	.	.	.	.	.	.
STY	.	.	3	4	.	4	.	.	.	.	.	.	.
TAX	.	.	.	.	.	.	.	.	2	.	.	.	.
TAY	.	.	.	.	.	.	.	.	2	.	.	.	.
TSX	.	.	.	.	.	.	.	.	2	.	.	.	.
TXA	.	.	.	.	.	.	.	.	2	.	.	.	.
TXS	.	.	.	.	.	.	.	.	2	.	.	.	.
TYA	.	.	.	.	.	.	.	.	2	.	.	.	.

* Sayfa sınırı boyunca indeksleniyorsa bir döngü ekleyin.
**Dallanma alınmışsa bir döngü ekleyin. Dallanma işlemi sayfa sınırını aşarsa bir döngü daha ekleyin.

00 – BRK
01 – ORA – (Indirect, X)
02 – Future Expansion
03 – Future Expansion
04 – Future Expansion
05 – ORA – Zero Page
06 – ASL – Zero Page
07 – Future Expansion
08 – PHP
09 – ORA – Immediate
0A – ASL – Accumulator
0B – Future Expansion
0C – Future Expansion
0D – ORA – Absolute
0E – ASL – Absolute
0F – Future Expansion
10 – BPL
11 – ORA – (Indirect), Y
12 – Future Expansion
13 – Future Expansion
14 – Future Expansion
15 – ORA – Zero Page, X

16 – ASL – Zero Page, X
17 – Future Expansion
18 – CLC
19 – ORA – Absolute, Y
1A – Future Expansion
1B – Future Expansion
1C – Future Expansion
1D – ORA – Absolute, X
1E – ASL – Absolute, X
1F – Future Expansion
20 – JSR
21 – AND – (Indirect, X)
22 – Future Expansion
23 – Future Expansion
24 – BIT – Zero Page
25 – AND – Zero Page
26 – ROL – Zero Page
27 – Future Expansion
28 – PLP
29 – AND – Immediate
2A – ROL – Accumulator
2B – Future Expansion
2C – BIT – Absolute
2D – AND – Absolute
2E – ROL – Absolute
2F – Future Expansion
30 – BMI
31 – AND – (Indirect), Y
32 – Future Expansion
33 – Future Expansion
34 – Future Expansion
35 – AND – Zero Page, X
36 – ROL – Zero Page, X
37 – Future Expansion
38 – SEC
39 – AND – Absolute, Y
3A – Future Expansion
3B – Future Expansion
3C – Future Expansion
3D – AND – Absolute, X
3E – ROL – Absolute, X
3F – Future Expansion
40 – RTI
41 – EOR – (Indirect, X)
42 – Future Expansion
43 – Future Expansion
44 – Future Expansion
45 – EOR – Zero Page
46 – LSR – Zero Page
47 – Future Expansion
48 – PHA
49 – EOR – Immediate
4A – LSR – Accumulator
4B – Future Expansion
4C – JMP – Absolute
4D – EOR – Absolute
4E – LSR – Absolute
4F – Future Expansion
50 – BVC
51 – EOR – (Indirect), Y
52 – Future Expansion
53 – Future Expansion
54 – Future Expansion
55 – EOR – Zero Page, X
56 – LSR – Zero Page, X
57 – Future Expansion
58 – CLI
59 – EOR – Absolute, Y
5A – Future Expansion
5B – Future Expansion
5C – Future Expansion
5D – EOR – Absolute, X
5E – LSR – Absolute, X
5F – Future Expansion
60 – RTS
61 – ADC – (Indirect, X)
62 – Future Expansion
63 – Future Expansion
64 – Future Expansion
65 – ADC – Zero Page
66 – ROR – Zero Page
67 – Future Expansion
68 – PLA
69 – ADC – Immediate
6A – ROR – Accumulator
6B – Future Expansion
6C – JMP – Indirect
6D – ADC – Absolute
6E – ROR – Absolute
6F – Future Expansion
70 – BVS
71 – ADC – (Indirect), Y

72 – Future Expansion
73 – Future Expansion
74 – Future Expansion
75 – ADC – Zero Page, X
76 – ROR – Zero Page, X
77 – Future Expansion
78 – SEI
79 – ADC – Absolute, Y
7A – Future Expansion
7B – Future Expansion
7C – Future Expansion
7D – ADC – Absolute, X
7E – ROR – Absolute, X
7F – Future Expansion
80 – Future Expansion
81 – STA – (Indirect, X)
82 – Future Expansion
83 – Future Expansion
84 – STY – Zero Page
85 – STA – Zero Page
86 – STX – Zero Page
87 – Future Expansion
88 – DEY
89 – Future Expansion
8A – TXA
8B – Future Expansion
8C – STY – Absolute
8D – STA – Absolute
8E – STX – Absolute
8F – Future Expansion
90 – BCC
91 – STA – (Indirect), Y
92 – Future Expansion
93 – Future Expansion
94 – STY – Zero Page, X
95 – STA – Zero Page, X
96 – STX – Zero Page, Y
97 – Future Expansion
98 – TYA
99 – STA – Absolute, Y
9A – TXS
9B – Future Expansion
9C – Future Expansion
9D – STA – Absolute, X
9E – Future Expansion
9F – Future Expansion
A0 – LDY – Immediate
A1 – LDA – (Indirect, X)
A2 – LDX – Immediate
A3 – Future Expansion
A4 – LDY – Zero Page
A5 – LDA – Zero Page
A6 – LDX – Zero Page
A7 – Future Expansion
A8 – TAY
A9 – LDA – Immediate
AA – TAX
AB – Future Expansion
AC – LDY – Absolute
AD – LDA – Absolute
AE – LDX – Absolute
AF – Future Expansion
B0 – BCS
B1 – LDA – (Indirect), Y
B2 – Future Expansion
B3 – Future Expansion
B4 – LDY – Zero Page, X
B5 – LDA – Zero Page, X
B6 – LDX – Zero Page, Y
B7 – Future Expansion
B8 – CLV
B9 – LDA – Absolute, Y
BA – TSX
BB – Future Expansion
BC – LDY – Absolute, X
BD – LDA – Absolute, X
BE – LDX – Absolute, Y
BF – Future Expansion
C0 – CPY – Immediate
C1 – CMP – (Indirect, X)
C2 – Future Expansion
C3 – Future Expansion
C4 – CPY – Zero Page
C5 – CMP – Zero Page
C6 – DEC – Zero Page
C7 – Future Expansion
C8 – INY
C9 – CMP – Immediate
CA – DEX
CB – Future Expansion
CC – CPY – Absolute
CD – CMP – Absolute

CE – DEC – Absolute
CF – Future Expansion
D0 – BNE
D1 – CMP – (Indirect), Y
D2 – Future Expansion
D3 – Future Expansion
D4 – Future Expansion
D5 – CMP – Zero Page, X
D6 – DEC – Zero Page, X
D7 – Future Expansion
D8 – CLD
D9 – CMP – Absolute, Y
DA – Future Expansion
DB – Future Expansion
DC – Future Expansion
DD – CMP – Absolute, X
DE – DEC – Absolute, X
DF – Future Expansion
E0 – CPX – Immediate
E1 – SBC – (Indirect, X)
E2 – Future Expansion
E3 – Future Expansion
E4 – CPX – Zero Page
E5 – SBC – Zero Page
E6 – INC – Zero Page
E7 – Future Expansion
E8 – INX
E9 – SBC – Immediate
EA – NOP
EB – Future Expansion
EC – CPX – Absolute
ED – SBC – Absolute
EE – INC – Absolute
EF – Future Expansion
F0 – BEQ
F1 – SBC – (Indirect), Y
F2 – Future Expansion
F3 – Future Expansion
F4 – Future Expansion
F5 – SBC – Zero Page, X
F6 – INC – Zero Page, X
F7 – Future Expansion
F8 – SED
F9 – SBC – Absolute, Y
FA – Future Expansion
FB – Future Expansion
FC – Future Expansion
FD – SBC – Absolute, X
FE – INC – Absolute, X
FF – Future Expansion

COMMODORE 64'ÜN BELLEK YÖNETİMİ

Commodore 64'ün 64K baytlık bir RAM'ı, bir de BASIC dilini, işletim sistemini ve standart karakter setini içeren 20K'lık ROM'u vardır. Aynı zamanda giriş/çıkış cihazlarına, belleğin 4K'lık bölümleriymiş gibi erişebilir. 16-bitlik adres bus'ı 64K için kullanılırken, bu işlem nasıl gerçekleşebiliyor?

Bu işin sırrı 6510 mikroişlemcisinin kendisindedir. Çipin üzerinde bir giriş/çıkış portu vardır. Bu port sistem belleğinin belirli kısımlarında RAM'ın mı, ROM'un mu yoksa giriş/çıkışın mı görüneceğini kontrol eder. Teybin kontrol edilmesinde de yine aynı port kullanılır. Önemli olan uygun bitlerin etkilenmesidir.

6510 giriş/çıkış portu belleğin 1'inci yerleşiminde yer alır. Veri yön kaydı ise 0'ıncı yerleşimdedir. Port, sistemdeki herhangi bir giriş/çıkış portu gibi kontrol edilir. Veri yön kaydı, verilen bir bitlik bilginin giriş mi, yoksa çıkış için mi kullanılacağını saptar. Böylece veri transferi portun kendi içinde gerçekleşir.

6510 kontrol portunun ana hatları aşağı da verilmiştir:

İSİM	BİT	YÖN	AÇIKLAMA
LORAM	0	ÇIKIŞ	$A000-$BFFF (BASIC) arasında yer alan RAM/ROM için kontrol
HIRAM	1	ÇIKIŞ	$E000-$FFFF (KERNAL) arasında yer alan RAM/ ROM için kontrol
CHAREN	2	ÇIKIŞ	$D000·$DFFF arasında yer alan Giriş/Çıkış ya da ROM için kontrol
	3	ÇIKIŞ	Kasete yazma hattı
	4	GİRİŞ	Teyp anahtarı duyumu
	5	ÇIKIŞ	Teyp motoru kontrolü

Veri yön kaydı için uygun olan değerler şöyle olabilir:

BİT	5	4	3	2	1	0
	1	0	1	1	1	1

(burada 1 çıkış için, 0 da giriş için kullanılmıştır.)

Değer ondalık olarak, 47'ye karşılık gelmektedir. Commodore 64 veri yön kaydını otomatik olarak bu değere ayarlar.

Genelde kontrol bağlantıları, açıklamalarında verilen fonksiyonları yerine getirirler. Fakat ara sıra özel bir bellek dağılımı elde etmek için bu kontrol hatlarının kombinasyonları kullanılır.

LORAM (bit 0) 8K baytlık BASIC ROM'unu kontrol eder. Yani, bu bölümde yer alan BASIC ROM'un, mikroişlemcinin adres alanından çıkartılmasını veya tekrar yerleştirilmesini sağlar. Bu hat genelde BASIC işlemler için ÜST (1) durumundadır. Bu hattı ALT (0) olarak programladığınızda BASIC ROM, bellek haritasındaki yerini 8K baytlık RAM'e bırakır. Yani bu işlemden sonra. $A000-$BFFF arasındaki yerleri makine dili programlarınız için kullanabilirsiniz.

HIRAM (bit 1) 8K baytlık KERNAL ROM'un mikroişlemcinin adres alanından çıkarılmasını ya da tekrar yerleştirilmesini kontrol eder.

Genelde bu hat. BASIC işlemleri için ÜST (1) olarak programlanmıştır. Bu hattı ALT (0) olarak programladığınızda KERNAL ROM, $E000-$FFFF adresleri arasındaki yerini 8K baytlık RAM'e bırakacaktır.

CHAREN (bit 2) 4K baytlık karakter üretici ROM'un mikroişlemcinin adres alanından çıkartılması ya da tekrar yerleştirilmesi işlemlerini kontrol eder. İşlemci açısından karakter ROM'u, giriş/çıkış cihazlarının da kullandığı $D000-$DFFF arasındaki alanı işgal eder. CHAREN hattına 1 yerleştirildiğinde (ki, normalde böyledir), giriş/çıkış cihazları mikroişlemcinin adres alanında yer alır ve karakter ROM artık erişilemez hale gelir. CHAREN bitine 0 yerleştirildiğinde ise karakter ROM buraya yerleşir ve artık giriş/çıkış cihazlarına erişilemez. (Mikroişlemci karakter ROM'una yalnızca karakter setinin ROM'dan RAM'e aktarılması sırasında gereksinim duyar. Bu işlemin ne olduğunu, grafikler bölümündeki programlanabilir karakterleri inceleyerek daha iyi anlayabilirsiniz). CHAREN belirli bellek konfigürasyonlarında diğer kontrol hatları tarafından kullanılabilir. Giriş/çıkış cihazları olmadan CHAREN'in diğer bellek konfigürasyonları üzerinde herhangi bir etkisi olmayacaktır. Bunun yerine $D000-$DFFF arasında RAM yer alacaktır.

NOT: ROM'un içerildiği herhangi bir bellek haritasında, ROM'a yazılmak (WRITE ya da POKE ile) istenilen bir veri, ROM'un altındaki RAM'de yer alacaktır. ROM'a yazıldığı sanılan veri, "gizli" RAM'de korunur. Örneğin: bu yüksek-çözünürlüklü bir ekranın ROM'un altına yerleştirilmesini ve ekranın, mikroişlemcinin adres alanına tekrar yerleştirilmesine gerek duyulmaksızın değiştirilebilmesini sağlar. Şüphesiz ROM'u okuduğunuzda elde edeceğiniz "gizli" RAM'ın değil ROM'un içeriğidir.

COMMODORE 64 TEMEL BELLEK HARİTASI:

Adres	İçerik
$E000-$FFFF	**8K KERNAL ROM VEYA RAM**
$D000-$DFFF	**4K G/Ç VEYA RAM VEYA KARAKTER ROM**
$C000-$CFFF	**4K RAM**
$A000-$BFFF	**8K BASIC ROM VEYA RAM VEYA ROM EKLENTİ**
$8000-$9FFF	**8K RAM VEYA ROM EKLENTİ**
$4000-$7FFF	**16K RAM**
$0000-$3FFF	**16K RAM**

GİRİŞ/ÇIKIŞ DAĞILIMI:

D000-D3FF VIC (Video kontrolörü)	1K Bayt
D400-D7FF SID (Ses synthesizer'ı)	1K Bayt
D800-DBFF Renk RAM'ı	1K Yarım-Bayt
DC00-DCFF CIA1 (Klavye)	256 Bayt
DD00-DDFF CIA2 (Seri bağlantı, Kullanıcı Portu/RS-232)	256 Bayt
DE00-DEFF Açık G/Ç slotu # 1 (CP/M kullanılabilir)	256 Bayt
DF00·DFFF Açık G/Ç slotu # 2 (Disk)	256 Bayt

İki tane açık giriş/çıkış slotu, kullanıcının genel amaçlı giriş/çıkış kullanımı, özel amaçlı girişi/çıkış kartuşları (IEEE gibi) ve Z·80 kartuşlarının kullanımını sağlamak amacıyla tasarlanmıştır.

Sistem, Commodore 64 Genişleme Kartuşu programlarının otomatik olarak çalışabilmelerini sağlar. 32768 ($8000) adresinden başlayan kartuş ROM'unun ilk dokuz baytında belirli veriler yer alıyorsa kartuş programı başlayacaktır. İlk iki baytta, kartuş programı tarafından kullanılacak Cold Start vektörü bulunmalıdır. Bunlardan sonraki üç bayt, 7'nci bitleri 1 olan CBM harfleri olmalıdır. Diğer iki bayt da PET ASCII'deki "80" sayıları olacaktır.

COMMODORE 64 BELLEK HARİTASI:

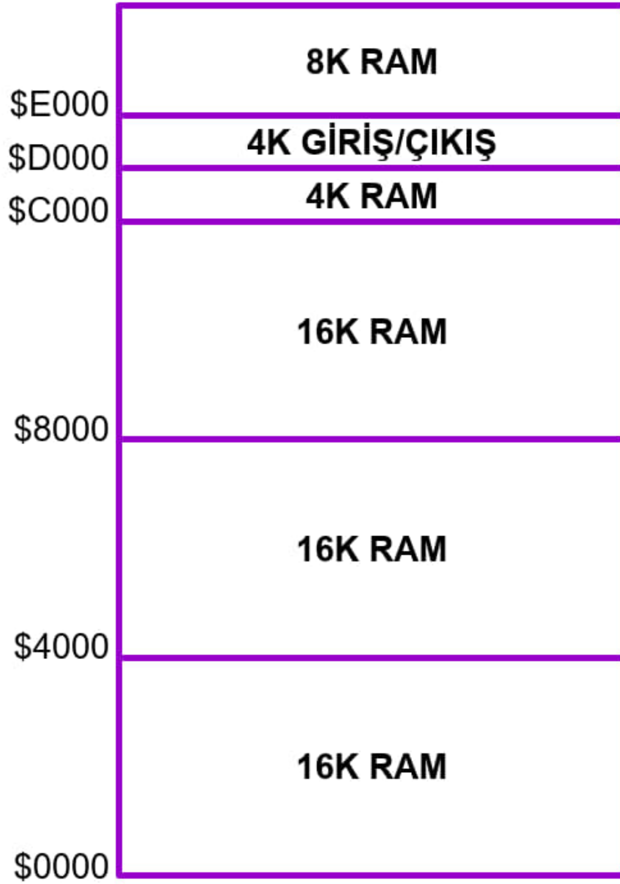
Aşağıdaki tablolarda COMMODORE 64'te bulunan çeşitli bellek konfigürasyonları, her bellek haritasını seçen kontrol hatlarının durumları ve her haritanın kullanım amacı listelenmektedir.

Adres	Bellek
	8K KERNAL ROM
$E000	4K GİRİŞ/ÇIKIŞ
$D000	4K RAM (TAMPON)
$C000	8K BASIC ROM
$A000	8K RAM
$8000	16K RAM
$4000	16K RAM
$0000	

X = ÖNEMLİ DEĞİL
0 = DÜŞÜK
1 = YÜKSEK

LORAM = 1
HIRAM = 1
GAME = 1
EXROM = 1

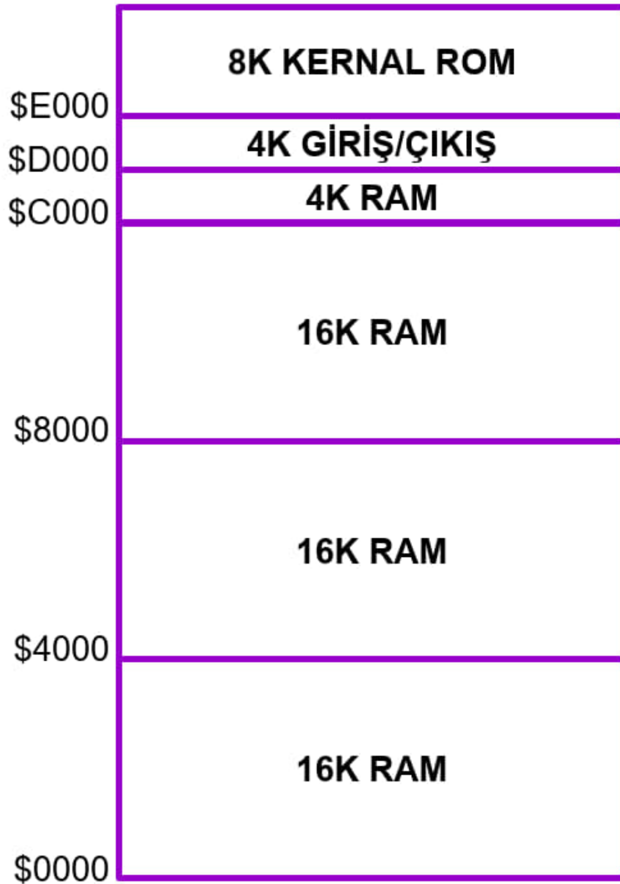
Bu, BASIC 2.0 ve 38K bitişik kullanıcı RAM baytı sağlayan varsayılan BASIC bellek haritasıdır.



X = ÖNEMLİ DEĞİL
0 = DÜŞÜK
1 = YÜKSEK

LORAM = 1
HIRAM = 0
GAME = 1
EXROM = X
Veya
LORAM = 1
HIRAM = 0
GAME = 0
(Karakter ROM'una bu haritadaki CPU tarafından erişilmemektedir.)
EXROM = 0

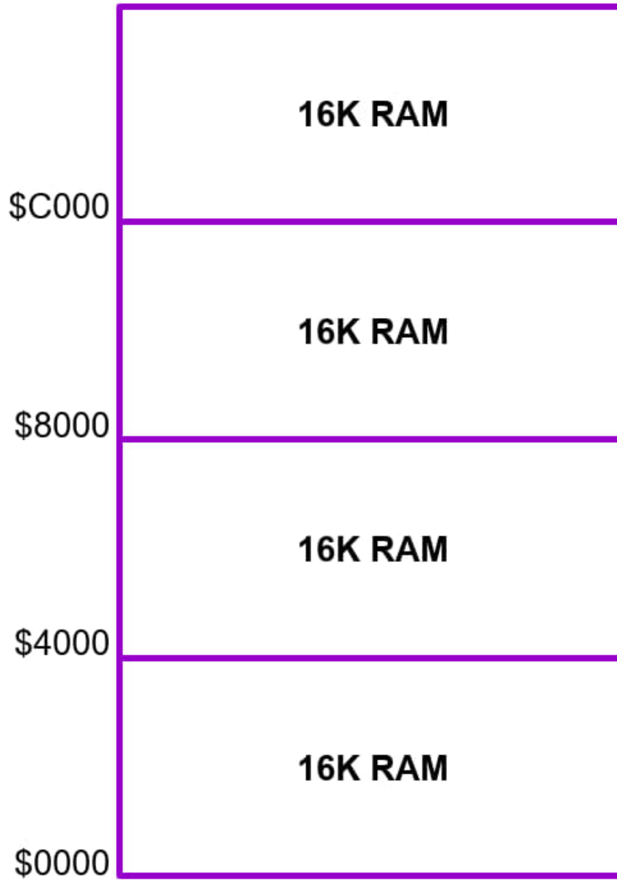
Bu harita 60K bayt RAM ve G/Ç aygıtları sağlar. Kullanıcının kendi G/Ç sürücüsü rutinlerini yazması gerekir.



X = ÖNEMLİ DEĞİL
0 = DÜŞÜK
1 = YÜKSEK

LORAM = 0
HIRAM = 1
GAME = 1
EXROM = X

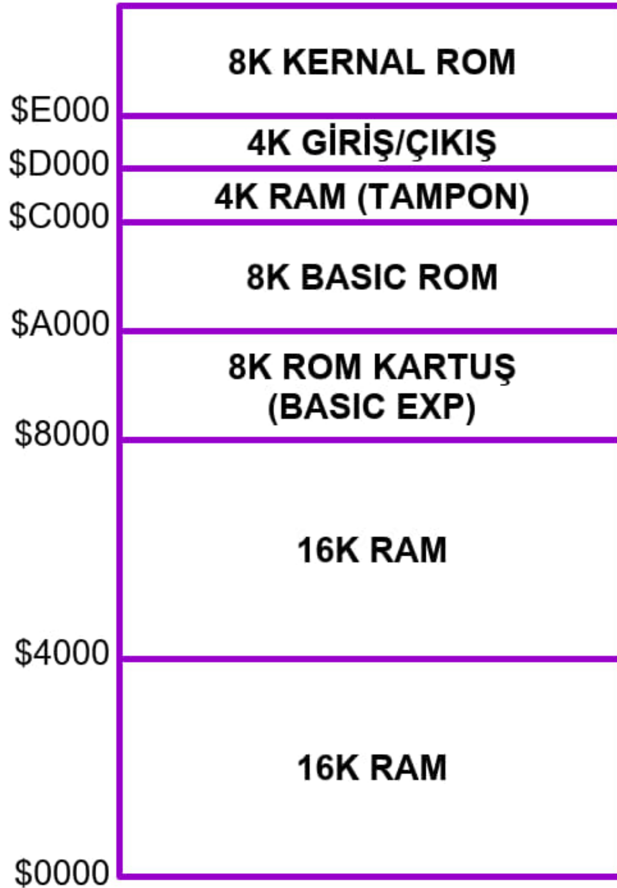
Bu harita, 52K bitişik Bayt kullanıcı RAM'i, G/Ç aygıtları ve G/Ç sürücüsü rutinleri sağlayan Softload dilleriyle (CP/M dahil) kullanılmak üzere tasarlanmıştır.



X = ÖNEMLİ DEĞİL
0 = DÜŞÜK
1 = YÜKSEK

LORAM = 0
HIRAM = 0
GAME = 1
EXROM = X
Veya
LORAM = 0
HIRAM = 0
GAME = X
EXROM = 0

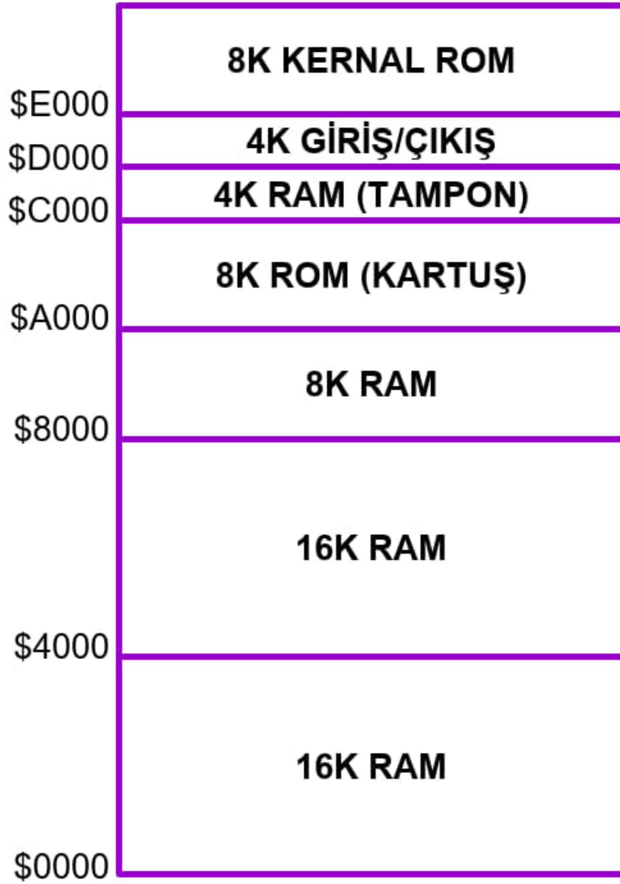
Bu harita 64K baytlık RAM'in tamamına erişim sağlar. Herhangi bir G/Ç işlemi için G/Ç cihazlarının işlemcinin adres alanına geri yerleştirilmesi gerekir.



X = ÖNEMLİ DEĞİL
0 = DÜŞÜK
1 = YÜKSEK

LORAM = 1
HIRAM = 1
GAME = 1
EXROM = 0

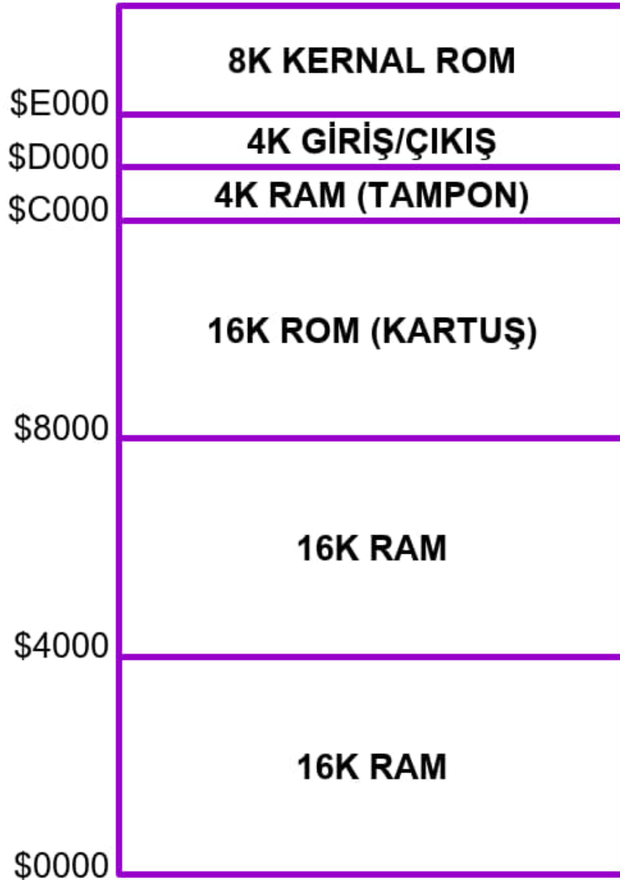
Bu, BASIC genişletme ROM'una sahip bir BASIC sistemi için standart konfigürasyondur. Bu harita, 32K bitişik bayt kullanıcı RAM'i ve 8K bayta kadar BASIC "geliştirme" sağlar.



X = ÖNEMLİ DEĞİL
0 = DÜŞÜK
1 = YÜKSEK

LORAM = 0
HIRAM = 1
GAME = 0
EXROM = 0

Bu harita, BASIC gerektirmeyen özel ROM tabanlı uygulamalar için 40K bitişik bayt kullanıcı RAM'i ve 8K bayta kadar eklenti ROM sağlar.



X = ÖNEMLİ DEĞİL
0 = DÜŞÜK
1 = YÜKSEK

LORAM = 1
HIRAM = 1
GAME = 0
EXROM = 0

Bu harita, BASIC gerektirmeyen özel ROM tabanlı uygulamalar (kelime işlemciler, diğer diller, vb.) için 32K bitişik bayt kullanıcı RAM'i ve 16K bayta kadar eklenti ROM sağlar.

Adres	Bölge
	8K KARTUŞ ROM
$E000	4K GİRİŞ/ÇIKIŞ
$D000	4K AÇIK
$C000	8K AÇIK
$A000	8K KARTUŞ ROM
$8000	16K AÇIK
$4000	12K RAM
$1000	4K RAM
$0000	

X = ÖNEMLİ DEĞİL
0 = DÜŞÜK
1 = YÜKSEK

LORAM = X
HIRAM = X
GAME = 0
EXROM = 1

Bu, ULTIMAX video oyunu hafıza haritasıdır. ULTIMAX için 2K baytlık "genişletme RAM'ine" gerekirse COMMODORE 64 üzerinden erişildiğini ve kartuştaki RAM'in göz ardı edildiğini unutmayın.

KERNAL

Programcıların, mikrobilgisayar alanında karşılaştıkları sorunlar arasında en çok düşündüreni, şirket tarafından bilgisayarın işletim sisteminde değişiklik yapıldığı zaman ne yapacaklarını bilmemeleridir. Büyük çabalar ve zamanlar harcayarak geliştirdikleri makine dili ile yazılmış programlar, bu değişikliklerden sonra çalışmamaktadır. Commodore, bu sorunları biraz olsun hafifletmek amacıyla, yazılım uzmanları için, KERNAL adı verilen bir metot geliştirdi.

Tam anlamıyla KERNAL**,** işletim sistemindeki giriş, çıkış ve bellek yönetimi programları için, standartlaştırılmış bir JUMP (SIÇRAMA) TABLOSU'dur. Sistemde bir değişiklik yapıldığında, bu programların ROM'daki yerleri de değişecektir. Fakat KERNAL sıçrama tabloları da bu değişikliğe göre yeniden düzenlenir. Eğer makine dili ile yazdığınız programlar, yalnızca KERNAL'daki ROM sistem programlarını kullanacaklarsa, bunların uyarlanması daha az zaman alacaktır.

KERNAL Commodore 64'un İşletim Sistemi'dir ve tüm giriş, çıkış ve bellek yönetimi işlemleri KERNAL tarafından kontrol edilir.

Makine dili ile yazdığınız programları basitleştirmek ve bu programların, Commodore 64'ün işletim sistemin in gelecekte geliştirildiği koşullarda yararsız hale geçmelerini önlemek için, KERNAL'da sizin kullanımınıza açık bir sıçrama tablosu vardır. Bu tabloda yer alan 39 giriş/çıkış programı ve diğer utility (yardımcı) programlarını kullanarak hem zamandan kazanacak hem de bu programları bir Commodore bilgisayarından diğerine aktarma olanağı bulacaksınız.

Sıçrama tablosu belleğin son sayfasına, ROM (Salt Okunur Bellek)'e yerleştirilmiştir.

KERNAL sıçrama tablosunu kullanabilmek için, öncelikle KERNAL programlarının kullanacağı değişkenleri tanımlamanız gereklidir. Bu işlemi bitirdikten sonra JSR (alt-programa atla) komutu ile, KERNAL tablosundaki uygun olan yere dallanabilirsiniz. İşlem bittikten sonra KERNAL, kontrolü tekrar makine dili programınıza devredecektir. Kullandığınız KERNAL altprogramına bağlı olarak, bazı kayıtlar değişkenleri programınıza geçirecektir. KERNAL alt-programlarında gerekli olan kayıtlar, açıklamaların yapıldığı bölümde her bir alt-program için ayrı ayrı belirtilmiştir.

Bu noktada akla gelebilecek en iyi soru, sıçrama tablolarının niçin kullanıldığı ve niçin programların doğrudan kullanılmasına yönelik JSR komutlarının kullanılmaması gerektiği... sıçrama tablosu, KERNAL ya da BASIC değişikliğe uğradığında, makine dili ile yazmış olduğunuz programların, bu değişiklik yüzünden kullanılmaz hale gelmesini önlemek amacıyla kullanılır. Yeni işletim sistemlerinde, programların bellek haritasındaki yerleri değişik olabilir, fakat sıçrama tablosu yine de doğru olarak çalışacaktır.

KERNAL'IN BAŞLANGIÇ AKTİVİTELERİ

1) Bilgisayar ilk açıldığında, KERNAL ilk olarak yığın göstergecini sıfırlar ve onlu (decimal) moddan çıkar.

2) Daha sonra, $8000 HEX (onlu sistemde 32768) yerleşimindeki, otomatik olarak çalışmaya başlayan ROM kartuşlarının takılı olup olmadığını kontrol eder. Eğer varsa, normal başlama işlemleri ertelenir ve kontrol, kartuş koduna transfer edilir. Yoksa, sistemin normal başlangıç işlemlerine devam edilir.

3) Sonra KERNAL, tüm giriş/çıkış cihazlarını ve seri bağlantıları, başlama durumuna getirir. 6526 CIA çipleri, klavyenin taranması (scanning) için belirli değerler alır, 60Hz zamanlayıcısı çalıştırılır. SID çipi sıfırlanır. BASIC bellek haritası seçilir ve kaset motorunun çalışması durdurulur.

4) KERNAL'ın bundan sonra yapacağı işlem, bellek göstericilerinin başını ve sonunu saptayan RAM testidir. Ayrıca, sıfırıncı sayfa başlama durumuna getirilir, teyp tamponu hazırlanır.

RAM testi $0300'den yukarı doğru işleyen zararsız bir testtir. Test sırasında, RAM'a ait olmayan ilk yerleşim bulunduğunda, RAM'in üst kısmını belirten gösterici tanımlanır. Belleğin sonu, her zaman $0800, ekran da daima $0400 olarak saptanır.

5) Son olarak, KERNAL şu işlemleri yapar: giriş/çıkış vektörlerinin default (başlangıç) değerlerini alması sağlanır. Belleğin alt bölümlerindeki, dolaylı sıçrama tabloları yerleştirilir. Ekran temizlenir ve tüm ekran editörü değişkenleri başlama durumuna getirilir. Daha sonra $A000'daki BASIC'i başlatmak için kullanılır.

KERNAL NASIL KULLANILIR?

Makine dili programların yazılmasında, sistem zamanlayıcısına erişim, bellek yönetimi, giriş/çıkış ve benzeri işlemler için, işletim sisteminin bir parçası olan bazı programların kullanılması gerekir. Bu programları tekrar tekrar yazmak gereksiz olduğundan, işletim sisteminin kolay erişilebilirliği, hızlı makine dili programcılığına olanak verir.

Daha önce de belirtildiği gibi, KERNAL bir sıçrama tablosudur. Yani, işletim sistemi alt-programları ile ilgili, JMP komutlarının seti olarak da adlandırılabilir.

Bir KERNAL alt-programını kullanabilmek için: önce o programın gereksindiği tüm hazırlıkları yerine getirmek gerekir. Eğer bir program, diğer bir KERNAL alt-programının çağırılmasını gerektirirse, çağırılması, akümülatöre bir sayı konulması istenirse, bu sayının konması gerekir. Bu işlemler yapılmadığı takdirde, programlarınızın beklediğiniz gibi çalışması çok düşük bir olasılıktır.

Tüm hazırlıklar yapıldıktan sonra, programı JSR yönergesiyle çağırmalısınız. Erişebileceğiniz bütün KERNAL programları, alt-programlar olarak yazıldığından RTS yönergesiyle bitmelidir. Bir KERNAL alt-programı görevini tamamladığında, kontrol, programınızdaki JSR yönergesinden sonraki yönergeye döner.

KERNAL programlarının çoğu, bir probleminiz olduğunda, statü sözcüğü (status word) veya akümülatörde hata kodu verir. Makine dili programlarınızın başarısı bu özelliğin iyi değerlendirilmesine bağlıdır. Hata mesajını ihmal ederseniz, programınızın diğer kısımları çalışmayabilir.

KERNAL kullanılırken uygulamanız gereken sadece üç şık vardır:

1) Yerleştirme (Setup)
2) Programı çağırma
3) Hata değerlendirmesi

Aşağıdaki terimler, KERNAL alt-programının açıklamalarında kullanılacaktır:

- **FONKSİYON İSMİ:** KERNAL alt-programının ismi.

- **ÇAĞRI ADRESİ:** KERNAL programının onaltılı sayı sistemindeki çağrılma adresi.

- **İLETİŞİM KAYITLARI:** Bu başlık altında verilen kayıtlar, değişkenleri KERNAL alt-programlarına göndermek veya onlardan almak için kullanılır.

- **HAZIRLIK PROGRAMLARI:** Bazı KERNAL programları için işlemeye başlamadan önce bazı verilerin hazır olması gerekir. Bu işlemleri yapan programlar bu başlık altında verilir.

- **HATA MESAJLARI:** Bir KERNAL programından taşıma bir yapılarak gönderilen hata mesajı, işlemde bir hata olduğunu gösterir. Hata no.'su akümülatörde içerilir.

- **YIĞIN GEREKSİNİMLERİ:** KERNAL programının içinde kullanılan yığın bayt sayısıdır.

- **ETKİLENEN KAYITLAR:** KERNAL programı tarafından kullanılan tüm kayıtlar burada toplanır.

- **TANIMLAMA:** Kullanılan KERNAL programının işlevi hakkında kısa bir açıklama verir.

KERNAL programlarının listesi aşağıda verilmiştir.

KULLANICI TARAFINDAN ÇAĞRILABİLEN KERNAL RUTİNLERİ

İSİM	ADRES		İŞLEVİ
	ONALTILIK	ONLUK	
ACPTR	$FFA5	65445	Seri port'dan bayt girişi.
CHKIN	$FFC6	65478	Bilgi girişi için kanal açmak.
CHKOUT	$FFC9	65481	Bilgi çıkışı için kanal açmak.
CHRIN	$FFCF	65487	Kanaldan karakter girişi.
CHROUT	$FFD2	65490	Kanaldan karakter çıkışı.
CIOUT	$FFA8	65448	Seri port'dan bayt çıkışı.
CINT	$FF81	65409	Ekran editörünü başlangıç durumuna getirmek.
CLALL	$FFE7	65511	Tüm dosya ve kanalları kapamak.
CLOSE	$FFC3	65475	Belirlenmiş mantıksal bir dosyayı kapatmak.
CLACHN	$FFCC	65484	Giriş/çıkış kanallarını kapatmak.
GETIN	$FFE4	65508	Klavye tamponundan karakter almak.
IOBASE	$FFF3	65523	Giriş/çıkış cihazları taban adreslerini saptamak.
IOINIT	$FF84	65412	Giriş/çıkış cihazları başlama durumuna getirmek.
LISTEN	$FFB1	65457	Seri bağlantılı tüm cihazları dinleme için uyarmak.
LOAD	$FFD5	65493	Cihazdan RAM'e yüklemek.
MEMBOT	$FF9C	65436	Bellek tabanını okuma/saptama.
MEMTOP	$FF99	65433	Bellek başlangıcını okuma/saptama.
OPEN	$FFC0	65472	Mantıksal bir dosya açmak.
PLOT	$FFF0	65520	Takipçinin X, Y konumunu okuma/saptama.
RAMTAS	$FF87	65415	RAM'i, teyp tomponu ve ekranı $0400'e ayarla.
RDTIM	$FFDE	65502	Gerçek zaman saatini okumak.
READST	$FFB7	65463	Giriş/çıkış durum sözcüğünü okumak.
RESTOR	$FF8A	65418	Giriş/çıkış vektörlerini eski konumuna getir.
SAVE	$FFD8	65496	RAM'in içeriğini cihaza kaydet.
SCNKEY	$FF9F	65439	Klavyeyi tara.
SCREEN	$FFED	65517	Ekranın X, Y organizasyonunu saptamak.
SECOND	$FF93	65427	Dinleden sonra ikincil adresi göndermek.
SETLFS	$FFBA	65466	Mantıksal, birinci, ikinci adresleri yerleştirmek.
SETMSG	$FF90	65424	KERNAL uyarılarını kontrol etmek.
SETNAM	$FFBD	65469	Dosya ismi vermek.
SETTIM	$FFDB	65499	Gerçek zaman saatini çalıştırmak.
SETTMO	$FFA2	65442	Seri bağlantıya çalışma aralığı vermek.
STOP	$FFE1	65505	Stop tuşunu kontrol etmek.
TALK	$FFB4	65460	Seri bağlantıya Konuş komutu vermek.
TKSA	$FF96	65430	Konuşdan sonra ikincil adresi göndermek.
UDTIM	$FFEA	65514	Gerçek zaman saatini arttırmak.
UNLSN	$FFAE	65454	Seri bağlantıya dinleme komutu vermek.
UNTLK	$FFAB	65451	Seri bağlantıya konuşma komutu vermek.
VECTOR	$FF8D	65421	Vektörlenmiş giriş/çıkışı okuma/saptama.

B-1.

Fonksiyon adı : ACPTR
Amaç : Seri porttan bir baytlık veri almak
Çağrı adresi : $FFA5 (onaltılık) 65445 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : TALK, TKSA
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : 13
Etkilenen kayıtlar : .A, .X

Tanımlama: Seri bağlanmış olan cihazdan (Örneğin: disk), bilgi almak için kullanılan bir rutindir. Bu rutin, seri bağlantıdan verinin bir baytın, tam el sıkışması (full handshaking) yöntemini kullanarak alır. Veri, akümülatöre yerleştirilir. Bu rutine hazırlık olarak ilk önce, seri bağlanmış olan cihaza, veri göndermesi için konuş (TALK) rutininin işleme sokulması gerekir. Eğer giriş cihazının ikincil bir konuma gereksinimi varsa, bu rutini çağrılmadan önce, TKSA KERNAL rutini ile bu komut gönderilmelidir. Hatalar, durum sözcüğünde (status word) yer alırlar. READST rutini de durum sözcüğünü okumak için kullanılır.

Kullanımı: Üç aşamadan oluşur.

1) Seri bağlantıdaki cihaza, Commodore 64'e veri göndermeye hazırlanması için komut ver. (TALK ve TKSA rutinlerini kullanın.)
2) Rutini çağır (JSR kullanarak).
3) Veriyi sakla veya kullan.

Örnek:

Veri yolundan (BUS) bir bayt almak:

```
JSR ACPTR
STA veri          ;Veri = Alınan değer
```

B-2.

Fonksiyon adı : CHKIN
Amaç : Bilgi girişi için kanal açmak
Çağrı adresi : $FFC6 (onaltılık) 65478 (onluk)
İletişim kayıtları : .X
Hazırlık rutinleri : (OPEN)
Hata mesajları :
Yığın gereksinimi : Yok
Etkilenen kayıtlar : .A, .X

Tanımlama: KERNAL OPEN rutini tarafından açılmış bulunan herhangi bir mantıksal dosya, CHKIN rutini ile, giriş kanalı olarak tanımlanabilir. Doğal olarak kanaldaki cihazın da giriş cihazı olması gerekir. Yoksa hata oluşacak ve rutinin işletimi durdurulacaktır.

Eğer veriyi klavyeden değil de başka bir yerden alıyorsanız, veri girişi için. CHRIN veya GETIN rutinlerinden önce, bu rutinin çağrılması gerekir. Klavyeden alacağınız bilgiyi kullanacaksanız ve başka hiçbir giriş kanalı açılmamışsa OPEN rutinini çağırmaya gerek yoktur.

Bu rutin seri porttaki bir cihaz ile kullanıldığında, konuş (TALK) adresini (OPEN rutininde belirtildiyse ikincil adresi de) otomatik olarak gönderir.

Kullanımı: Üç aşamadan oluşur.

1) OPEN rutinini kullanarak bir dosya aç (gerekirse yukarıdaki açıklamaya bakınız).

2) .X kaydına, kullanılacak dosya numarasını yükle.

3) Bu rutini çağır (JSR komutu ile).

Olabilecek hatalar:

#3: File not open (Dosya açık değil)

#5: Device not present (Cihaz hazır değil)

#6: File not an input file (Dosya giriş dosyası değil)

Örnek:

Dosya 2'den giriş yaptırmak:

```
LDX #2
JSR CHKIN
```

B-3.

Fonksiyon adı : CHKOUT
Amaç : Çıkış için kanal açmak
Çağrı adresi : $FFC9 (onaltılık) 65481 (onluk)
İletişim kayıtları : .X
Hazırlık rutinleri : (OPEN)
Hata mesajları : 0, 3, 5, 7 READST'ye bakınız
Yığın gereksinimi : 4+
Etkilenen kayıtlar : .A, .X

Tanımlama: KERNAL OPEN rutini tarafından açılmış olan herhangi bir dosya numarası, çıkış kanalı olarak tanımlanabilir. Doğal olarak, kendisi için kanal açılması tasarlanan cihazın, çıkış cihazı olması gerekir. Çıkış cihazı değilse hata oluşacaktır ve rutin çalışmayacaktır.

Commodore 64'ün ekranını çıkış cihazı olarak kullanmak istemediğiniz sürece. Diğer cihazlar için, veri gönderilmeden evvel bu rutinin çağrılması gerekir. Ekrandan çıkış almak istenildiğinde ve halen tanımlanmış başka bir çıkış kanalı yoksa, OPEN rutinini çağırmaya gerek duymadan bu rutini çağırabilirsiniz.

Seri bağlantıdaki bir cihaz için kanal oluşturmak amacıyla kullanıldığında bu rutin, OPEN rutini ile belirtilen dinle (LISTEN) adresini otomatik olarak gönderecektir (ikincil adres varsa, o da gönderilecektir).

Kullanımı: Üç aşamadan oluşur.

> **HATIRLATMA:** Ekrana veri göndermek için bu rutinin kullanılmasına gerek yoktur.

1) KERNAL OPEN rutinini kullanarak, bir dosya numarası, bir dinleme (LISTEN) adresi ve ikincil adres (eğer gerekiyorsa) belirle.

2) Open yönergesinde kullanılan dosya numarasını .X kaydına yükle.

3) Rutini çağır (JSR komutu ile).

Olabilecek hatalar:
#3: File not open (Dosya açık değil)
#5: Device not present (Cihaz hazır değil)
#7: Not an output file (Çıkış dosyası değil)

Örnek:
LDX #3 ;Dosya 3'ü çıkış kanalı olarak tanımla
JSR CHKOUT

B-4.

Fonksiyon adı : CHRIN
Amaç : Giriş kanalından bir bayt almak
Çağrı adresi : $FFCF (onaltılık) 65487 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : (OPEN, CHKIN)
Hata mesajları : 0 (READST'ye bakınız)
Yığın gereksinimi : 7+
Etkilenen kayıtlar : .A, .X

Tanımlama: Bu rutin, KERNAL CHKIN rutini tarafından tanımlanan giriş kanalından, bir baytlık veri alınmasını sağlar. Eğer CHKIN rutini başka bir giriş kanalını tanımlamak için kullanılmadıysa, bütün veri girişinin klavyeden yapılması beklenir. Veri baytı akümülatörde yer alır ve kanal çağrıdan sonra açık kalır.

Klavyeden giriş, özel bir yolla yapılır. Önce takipçi ortaya çıkar ve Return'e basılıncaya kadar yanıp söner. Satırdaki tüm karakterler (88 karaktere kadar), BASIC giriş tamponuna saklanır. Bu karakterler, rutinin her bir karakter için çağrılmasıyla, bir defada bir tanesi okunarak elde edilirler. Return işareti algılandığında, tüm satır işlenir. Bu rutin sonradan çağrıldığında bütün bu işlemler, takipçinin yanıp sönmesiyle yeniden başlar.

Kullanımı: Klavyeden veri almada dört, başka cihazlardan veri almada üç aşamadan oluşur.

Başka cihazlardan;

1) KERNAL OPEN ve CHKIN rutinlerini kullan.
2) Bu rutini çağır (JSR kullanarak.)
3) Veriyi sakla

Klavyeden;

1) Rutini kullanarak bir baytlık veri al.
2) Veri baytın, sakla.
3) Son veri baytının Return olup olmadığını kontrol et.
4) Değilse birinci şıkka git.

Örnek:
Başka cihazlardan:
JSR CHKIN
STA Veri ;Veri = Alınan değer

Örnek:

Klavyeden:

```
     LDY $#00       ;.Y kaydını veriyi saklaması için hazırla
RD   JSR CHRIN
     STA Veri,Y     ;Verinin Y'ninci baytını, veri alanının Y'ninci yerleşiminde sakla
     INY
     CMP #CR        ;RETURN olup olmadığını kontrol et
     BNE RD         ;Değilse başka bir bayt al
```

B-5.

Fonksiyon adı : CHROUT
Amaç : Çıkış kanalına bir bayt gönder
Çağrı adresi : $FFD2 (onaltılık) 65490 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : (CHKOUT,OPEN)
Hata mesajları : 0 (READST'ye bakınız)
Yığın gereksinimi : 8+
Etkilenen kayıtlar : .A

Tanımlama: Bu rutin, önceden açılmış bir kanala bir bayt göndermek için kullanılır. Çağrılmadan evvel kanalın açılması için KERNAL OPEN ve CHKOUT rutinlerinin kullanılması gerekir. Çağrının unutulduğu durumlarda veri, varsayılan çıkış cihazına gönderilir. Bu da 3 numaralı cihaz olan ekrandır. Gönderilecek olan veri baytı önce akümülatöre yüklenir, sonra rutin çağrılır ve veri belirlenen çıkış cihazına gönderilir. Çağrıdan sonra bu kanal açık kalır.

NOT: Seri porttaki bir cihaza veri gönderirken, veri, porta bağlı açık olan tüm çıkış kanallarına gideceğinden, bu rutini çok dikkatli bir şekilde kullanmak gerekir Özel haller dışında, verinin gönderilmesini islediğimiz kanalın dışındaki tüm açık çıkış kanallarının KERNAL CLRCHN rutinini kullanarak kapatmamız gerekir.

Kullanımı: Üç aşamadan oluşur.

1) Gerekiyorsa CHKOUT KERNAL rutinini kullanın (yukarıdaki açıklamayı okuyun).
2) Gönderilecek veriyi akümülatöre yükleyin.
3) Rutini çağırın.

Örnek:

BASIC'teki (CMD 4, "A") komutunun eşdeğeri:

```
LDX #4          ;Mantıksal dosya #4
JSR CHKOUT      ;Çıkış kanalı aç
LDA #'A
JSR CHROUT      ;Karakter gönder
```

B-6.

Fonksiyon adı : CIOUT
Amaç : Seri porttaki bir cihaza bir bayt göndermek
Çağrı adresi : $FFA8 (onaltılık) 65448 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : LISTEN, (SECOND)
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : 5
Etkilenen kayıtlar : YOK

Tanımlama: Seri porttaki bir cihaza bilgi göndermek için kullanılır. Bu rutinin çağırılması sonucunda seri bağlantıya, tam el sıkışması (full handshaking) yöntemi kullanılarak bir baytlık veri gönderilir. Bu rutini çağırmadan önce, KERNAL dinle (LISTEN) rutinini kullanarak cihaza, veriyi almak için hazır olması gerektiği bildirilir. (Cihazın ikinci bir adrese gereksinimi varsa, KERNAL SECOND rutini ile bu adresin gönderilmesi gerekir). Akümülatöre, tam anlaşma için bir bayt yüklemelidir. Cihazın LISTEN konumunda olması gerekir; aksi halde statü sözcüğü timeout hatası gönderecektir. Bu rutin, tamponda daima bir bayt tutar. KERNAL UNLSN rutinine çağrı yapıldığında, önce tampona atılmış karakter "End or Identify (EOI)" ile birlikte gönderilir. UNLSN komutu cihaza bundan sonra gönderilir.

Kullanımı: Üç aşamadan oluşur.

1) LISTEN KERNAL rutinini kullanın (gerekirse SECOND rutinini de kullanmalısınız).
2) Akümülatöre bir baytlık veri yükleyin.
3) Rutini çağırın ve veri baytını gönderin.

Örnek:

```
LDA #'X         ;Seri porttaki cihaza bir X gönder
JSR CIOUT
```

B-7.

Fonksiyon adı : CINT
Amaç : Ekran editörü, 6567 video çipi başlama durumuna getirme
Çağrı adresi : $FF81 (onaltılık) 65409 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 4
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, Commodore 64'ün içindeki 6567 video kontrol çipini normal işlemler için hazırlar ve KERNAL ekran editörünü başlangıç durumuna getirir. Self-Start program kartuşlarında bu rutinin çağırılması gerekir.

Kullanımı: Bir aşamadan oluşur.

1) Rutini çağırın.

Örnek:

```
JSR CINT
JMP RUN             ;Çalışmaya başla
```

B-8.

Fonksiyon adı : CLALL
Amaç : Tüm dosyaları kapatmak
Çağrı adresi : $FFE7 (onaltılık) 65511 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 11
Etkilenen kayıtlar : .A, .X

Tanımlama: Bu rutin, açık olan tüm dosyaları kapatır. Çağırıldığında, açık dosya tablosundaki tüm göstergeçleri bütün dosyalar kapatılarak sıfırlanır. Ayrıca giriş/çıkış kanallarını sıfırlamak için CLRCHN rutini, otomatik olarak çağırılır.

Kullanımı: Bir aşamadan oluşur.

1) Rutini çağırın.

Örnek:

```
JSR CLALL           ;Tüm dosyaları kapat ve varsayılan g/ç kanallarını seç
JMP RUN             ;Çalışmaya başla
```

B-9.

Fonksiyon adı : CLOSE
Amaç : Bir dosyayı kapatmak
Çağrı adresi : $FFC3 (onaltılık) 65475 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : Yok
Hata mesajları : 0, 240 (READST'ye bakınız)
Yığın gereksinimi : 2+
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin bir dosyayı, üzerindeki tüm giriş/çıkış işlemleri tamamlandıktan sonra kapatmak için kullanılır. Kapatılacak olan dosyanın numarası akümülatöre yüklendikten sonra, bu rutin çağırılır. (Dosya numarası, dosyayı açmak için kullanılan OPEN rutinindeki numara ile aynı olmak zorundadır).

Kullanımı: İki aşamadan oluşur.

1) Akümülatöre kapatılacak dosyanın numarasını yükleyin.
2) Rutini çağırın.

Örnek:

15 numaralı dosyayı kapatmak:

```
LDA #15
JSR CLOSE
```

B-10.

Fonksiyon adı : CLRCHN
Amaç : Giriş/çıkış kanallarını temizlemek
Çağrı adresi : $FFCC (onaltılık) 65484 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : Yok
Hata mesajları :
Yığın gereksinimi : 9
Etkilenen kayıtlar : .A, .X

Tanımlama: Bu rutin, tüm açık kanalları temizlemek ve g/ç kanallarına orijinal varsayılan değerlerini yüklemek için kullanılır. Genellikle giriş/çıkış kanalları açıldıktan (teyp veya disk sürücü gibi) ve bu kanallar giriş/çıkış işlemleri için kullanıldıktan sonra çağırılır. Varsayılan çıkış cihazı ise, 3 no.'lu cihaz olan ekrandır.

Seri porttaki kanallardan biri kapatılacaksa, giriş kanalını sıfırlamak için rutin önce, bir konuşma (UNTALK) sinyali veya çıkış kanalını temizlemek için bir dinleme (UNLISTEN) sinyali gönderir. Bu rutin kullanılmadığı, dolayısıyla seri bağlantıdaki dinleyiciler işler durumda bırakıldığı zaman, birçok cihaz Commodore 64'ten aynı anda veri alabilir. Bu özellikten faydalanmanın bir yolu, yazıcıya konuş (TALK), diske de dinle (LISTEN) göndermektir. Bu işlem bir disk dosyasının doğrudan yazıcıya gönderilebilmesini sağlar.

Kullanımı: Bir aşamadan oluşur.

1) JSR komutunu kullanarak bu rutini çağırın.

Örnek:

```
JSR CLRCHN
```

B-11.

Fonksiyon adı : GETIN
Amaç : Karakter almak
Çağrı adresi : $FFE4 (onaltılık) 65508 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : CHKIN, OPEN
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : 7+
Etkilenen kayıtlar : .A (.X, .Y)

Tanımlama: Eğer kanal klavye ise, bu rutin klavye kuyruğundan (queue) bir karakter alır ve ASCII değerini akümülatöre yükler. Kuyrukta bir şey yoksa akümülatördeki değer sıfır olacaktır. Karakterler kuyruğa, SCNKEY rutinini çağıran interrupt klavye tarama rutini tarafından otomatik olarak yerleştirilir. Klavye tamponunda 10 tane karakter bulundurmak mümkündür. Tampon dolduğunda en az bir karakter tampondan çekilinceye kadar başka karakter girilemez. Eğer kanal RS-232 ise, sadece .A kaydı kullanılır ve tek bir karakter alınabilir. Doğruluğunu READST ile kontrol ediniz. Eğer kanal seri, kaset ya da ekran ise BASIN rutinini çağırın.

Kullanımı: Üç aşamadan oluşur.
1) JSR komutunu kullanarak rutini çağırın.
2) Akümülatörde sıfır olup olmadığını kontrol edin (boş tampon).
3) Veriyi işleyin.

Örnek:
Bir karakter için beklemek:

```
WAIT JSR GETIN
CMP #0
BEQ WAIT
```

B-12.

Fonksiyon adı : IOBASE
Amaç : Giriş/çıkış bellek sayfası tanımlamak
Çağrı adresi : $FFF3 (onaltılık) 65523 (onluk)
İletişim kayıtları : .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları :
Yığın gereksinimi : 2
Etkilenen kayıtlar : .X, .Y

Tanımlama: Bu rutin; VIC, SID, CIA gibi bellek adresli giriş/çıkış cihazlarının başlangıç adresini, X ve Y kayıtlarına yükler. Bu adres daha sonra bir ofset aracılığıyla, Commodore 64 içindeki giriş/çıkış cihazlarına erişmek için kullanılır. Ofset değeri, ulaşmak istediğiniz kayıtların giriş/çıkış kayıtlarının bulunduğu sayfanın başlangıcından kaç bayt ötede olduğunu gösterir. Adresin alt baytı .X kaydında, üst baytı ise .Y kaydında yer alır.

Bu rutin, Commodore 64, VIC·20 ve Commodore 64 'ün ileride çıkacak modelleri arasında uyum sağlamak amacıyla oluşturulmuştur. Eğer bir makine dili program için giriş/çıkış yerleşimleri bu rutin çağrılarak tanımlanmışsa, Commodore 64, KERNAL ve BASIC'ın gelecekteki çeşitleri için uyarlanabilir kalacaktır.

Kullanımı: Dört aşamadan oluşur.
1) JSR komutu ile rutini çağırın.
2) X ve Y kayıtlarını arka arkaya gelen iki yerleşime koyun.
3) Y kaydına ofseti yükleyin.
4) İstenilen giriş/çıkış yerleşimine erişin.

Örnek:
Kullanıcı portunun veri yön kaydına 0 (INPUT) yüklemek:

```
JSR IOBASE
STX POINT
STY POINT +1
LDY #2
LDA #0
STA (POINT), Y      ;Veri yön kaydına 0 yükle
```

B-13.

Fonksiyon adı : IOINIT
Amaç : Giriş/çıkış cihazlarını başlama durumuna getirmek
Çağrı adresi : $FF84 (onaltılık) 65412 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : Yok
Hata mesajları :
Yığın gereksinimi : Yok
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, tüm giriş/çıkış cihazlarını ve rutinlerini başlangıçtaki durumlarına dönüştürür. Normal olarak, Self-Start program kartuşlarında kullanılır.

Örnek:

```
JSR IOINIT
```

B-14.

Fonksiyon adı : LISTEN
Amaç : Seri bağlanmış bir cihaza "dinle" komutu göndermek
Çağrı adresi : $FFB1 (onaltılık) 65457 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : Yok
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : Yok
Etkilenen kayıtlar : .A

Tanımlama: Bu rutin, seri porttaki bir cihaza, veri almaya başlamasını bildirir. Bu rutini çağırmadan önce, akümülatöre 0 ile 31 arasında bir cihaz numarası yüklemek gerekir. Dinle (LISTEN) bu sayıyı, dinleme adresine dönüştürmek için bit-bit VEYA (OR) işlemine sokar ve bu veriyi, seri bağlantıya bir komut olarak gönderir. Belirlenen cihaz dinleme moduna girer ve gönderilecek bilgiyi almak için hazır duruma geçer.

Kullanımı: İki aşamadan oluşur.

1) Cihaza dinle (LISTEN) komutu vermek için, akümülatöre cihaz no yükleyin.
2) JSR komutu ile rutini çağırın.

Örnek:

```
LDA #8
JSR LISTEN          ;8 No.'lu cihaza dinle komutu ver
```

B-15.

Fonksiyon adı : LOAD
Amaç : Cihazdan RAM'e yüklemek
Çağrı adresi : $FFD5 (onaltılık) 65493 (onluk)
İletişim kayıtları : .A, .X, .Y
Hazırlık rutinleri : SETLFS, SETNAM
Hata mesajları : 0, 4, 5, 8, 9, READST
Yığın gereksinimi : Yok
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, herhangi bir giriş cihazından Commodore 64'ün belleğine veri baytlarını yüklemek amacıyla kullanılır. Aynı rutin, bellekteki verilerle cihazdaki verilerin aynı olup olmadığını kontrol eden VERIFY işlemi için de kullanılır.

Akümülatörün (.A), LOAD işlemi için 0, doğrulama işlemi için ise 1'le yüklenmesi gerekir. Eğer giriş cihazı, ikincil adres olarak 0 ile OPEN edilmişse, verilerin RAM'in neresine yükleneceğini gösteren başlık bilgileri okunmaz. Bu durumda X ve Y kayıtlarının yükleme için başlangıç adresini belirtmeleri gerekir. Cihaz, ikincil adres olarak 1 ile açılmışsa veri, belleğe, başlığın gösterdiği yerden itibaren yüklenmeye başlar. Rutin dönüşünde X ve Y kayıtları, yüklenen en son RAM yerleşiminin adresini verirler.

Bu rutinden önce, KERNAL SETLFS ve SETNAM rutinlerinin çağırılması gerekir.

NOT: Klavyeden (0), RS·232'den (2) ve ekrandan (3) yükleme yapılamaz.

Kullanımı: Dörti aşamadan oluşur.

1) SETLFS ve SETNAM rutinlerini çağırın. Eğer yükleme adresini değiştirmek istiyorsanız SETLFS rutininde ikincil adres olarak 0 kullanın.
2) .A'ya yükleme için 0, doğrulama için 1 değerini yükleyin.
3) Başlıkta gösterilenden farklı bir adrese yükleme yapılmasını istiyorsanız, .X ve .Y kayıtlarına başlangıç adresini yükleyin.
4) JSR komutu ile rutini çağırın.

Örnek:

Teypten bir dosya yüklemek:

```
LDA #cihaz1         ;Cihaz no tanımla
LDX #dosyano        ;Dosya no tanımla
LDY #ikincil        ;İkincil adresi tanımla
JSR SETLFS
LDA #isimuzunlugu   ;.A'ya dosya adının uzunluğunu yükle
LDX #isimalt        ;.X ve .Y'ye dosya adının başlangıç adresini yükle
LDY #isimüst
JSR SETNAM
LDA #0              ;yükleme için ikincil adres 1 ise x ve y değerleri fark etmez
LDX #$FF
LDY #$FF
JSR LOAD
STX VARTAB          ;yükleme sonu adresini kaydedebilirsiniz
STY VARTAB +1
JMP START
```

B-16.

Fonksiyon adı : MEMBOT
Amaç : Belleğin son kısmını tanımlamak
Çağrı adresi : $FF9C (onaltılık) 65436 (onluk)
İletişim kayıtları : .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : Yok
Etkilenen kayıtlar : .X, .Y

Tanımlama: Bu rutin, belleğin son kısmını öğrenmek ya da tanımlamak için kullanılır. Rutin çağırıldığında, akümülatörün taşıma biti 1 ise, .X ve .Y kayıtları RAM'in en alt baytının adresini verir. Genişletilmemiş bellekli Commodore 64'lerde bu adres $0800'dür (onlu sistemde 2048). Akümülatörün taşıma biti 0 iken çağırıldığında ise .X ve .Y kayıtlarına önceden verilmiş olan değerler, RAM'in başlangıç adresini tanımlar.

Kullanımı: İki bölüm ve her bölüm iki aşamadan oluşur.

RAM'in son kısmını okumak için:

1) Taşımayı 1 yapın.
2) Rutini çağırın.

RAM'in son kısmını tanımlamak için:

1) Taşımayı sıfırlayın.
2) Rutini çağırın.

Örnek:

Bellek son kısmını 1 sayfa yukarı taşımak:

```
SEC            ;Bellek son kısmını oku
JSR MEMBOT
INY
CLC            ;Bellek son kısmını yeni değere hazırlayın
JSR MEMBOT
```

B-17.

Fonksiyon adı : MEMTOP
Amaç : Belleğin ilk kısmını tanımlamak
Çağrı adresi : $FF99 (onaltılık) 65433 (onluk)
İletişim kayıtları : .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .X, .Y

Tanımlama: Bu rutin, RAM'ın ilk kısmını okumak ya da tanımlamak için kullanılır. Akümülatörün taşıma biti 1 iken çağırıldığında, RAM'in sonunu .X ve .Y kayıtlarına kaydeder. Taşıma biti 0 iken çağırıldığında ise, .X ve .Y kayıtlarında bulunan değerler belleğin ilk kısım adresine tanımlar.

Örnek:
RS-232 tamponunu kaldırmak:

```
SEC
JSR MEMTOP      ;Bellek ilk kısmını oku
DEX
CLC
JSR MEMTOP      ;Belleğin yeni ilk kısmını ayarla
```

B-18.

Fonksiyon adı : OPEN
Amaç : Bir dosya açmak
Çağrı adresi : $FFC0 (onaltılık) 65472 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : SETLFS, SETNAM
Hata mesajları : 1, 2, 4, 5, 6, 240, READST
Yığın gereksinimi : Yok
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin mantıksal bir dosya açmak için kullanılır. Açılan dosya, giriş/çıkış işlemleri için kullanılabilir ve KERNAL g/ç rutinlerinin çoğunda dosya açmak için bu rutin çağırılır. Kullanılması için bir parametre göndermeye gerek yoktur, fakat önceden SETLFS ve SETNAM KERNAL rutinlerinin çağırılmış olması gerekir.

Kullanımı: Üç aşamadan oluşur.
1) SETLFS rutinini ve,
2) SETNAM rutinini kullanın.
2) Bu rutini çağırın.

Örnek:
Aşağıda, OPEN 15, 8, 15, “I/O” BASIC yönergesinin bir uygulaması verilmiştir:

```
        LDA #isim2-isim  ;SETLFS için dosya adı uzunluğu
        LDY #isim        ;Dosya isminin adresleri
        LDX #isim
        JSR SETNAM
        LDA #15
        LDX #8
        LDY #15
        JSR SETLFS
        JSR OPEN
isim    .BYT 'G/Ç'
isim2
```

B-19.

Fonksiyon adı : PLOT
Amaç : Takipçinin yerini belirlemek
Çağrı adresi : $FFF0 (onaltılık) 65520 (onluk)
İletişim kayıtları : .A, .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Akümülatörün taşıma biti 1 iken bu rutin çağırıldığında, o anda takipçinin ekran üzerinde bulunduğu konum (X ve Y koordinatları olarak) .Y ve .X kayıtlarına kaydedilir. Y, takipçinin bulunduğu yerleşimin sütun numarası (0-39), X ise satır numarasıdır (0-24). Taşıma biti 0 iken çağrı yapıldığında takipçi, .Y ve .X kayıtlarında belirtilen X ve Y konumuna getirilir.

Kullanımı: İki bölüm ve her bölüm üç aşamadan oluşur.

Takipçinin yerini saptama:

1) Taşıma bitini 1 yapın.
2) Rutini çağırın.
3) X ve Y konumunu .Y ve .X kayıtlarından alın.

Takipçiyi yerleştirme:

1) Taşıma bitini sıfırlayın.
2) .Y ve .X kayıtlarına, takipçinin konun değerlerini yükleyin.
3) Rutini çağırın.

Örnek:

Takipçiyi 10'uncu satır, 5'inci sütuna getirmek (5, 10):

```
LDX #10
LDY #5
CLC
JSR PLOT
```

B-20.

Fonksiyon adı : RAMTAS
Amaç : RAM'i kontrol etmek
Çağrı adresi : $FF87 (onaltılık) 65415 (onluk)
İletişim kayıtları : .A, .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, RAM'i test etmek ve belleğin başlangıç ve bitimini göstericilere kaydetmek için kullanılır. Ayrıca $0000'dan $0101'e ve $0200'den $03FF'e kadar olan yerleşimleri temizler, kaset tamponunun yerini saptar, ekran tabanını $0400'e ayarlar. Normal olarak bu rutin, Self-Start program kartuşlarının başlangıç işlemlerinin bir bölümü olarak kullanılır.

Örnek:

```
JSR RAMTAS
```

B-21.

Fonksiyon adı : RDTIM
Amaç : Sistem saatinin okunması
Çağrı adresi : $FFDE (onaltılık) 65502 (onluk)
İletişim kayıtları : .A, .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, sistem saatinin okunmasında kullanılır. Saatin hassasiyeti saniyenin 60'da biridir. Bu rutin işleme sokulduktan sonra, 3 baytlık değerin en üst baytı akümülatörde, bir sonra gelen baytı X indeksinde, en alt baytı ise Y indeksinde yer alır.

Örnek:

```
JSR RDTIM
STY TIME
STX TIME+1
STA TIME+2
...
TIME *=*+3
```

B-22.

Fonksiyon adı : READST
Amaç : Durum sözcüğünün (status word) okunması
Çağrı adresi : $FFB7 (onaltılık) 65463 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A

Tanımlama: READST rutini, giriş/çıkış cihazlarının o andaki durumlarını akümülatöre kaydeder ve genellikle giriş/çıkış cihazı ile yeni bir iletişim kuruduğunda çağırılır. Bu rutin cihazın durumu veya Giriş/Çıkış işlemleri süresinde oluşan hatalar ile ilgili bilgi verir. Akümülatördeki bitlerin içerdiği bilgiler aşağıdaki tabloda verilmiştir:

ST BİT KONUMU	ST SAYISAL DEĞERİ	KASET OKUMA	SERİ VERİ YOLU OKUMA/YAZMA	TEYP KONTROL + YÜKLEME
0	1		Yazma zaman aşımı	
1	2		Okuma zaman aşımı	
2	4	Kısa blok		Kısa blok
3	8	Uzun blok		Uzun blok
4	16	Okuma hatası		Herhangi bir hata
5	32	Sağlama hatası		Sağlama hatası
6	64	Dosya sonu	EOI sinyali	
7	-128	Teyp sonu	Cihaz hazır değil	Teyp sonu

Kullanımı: İki aşamadan oluşur.
1) Rutini çağırın.
2) .A kaydında yer alan bilginin ne olduğunu anlayın.

Örnek:
Okuma sırasında dosya sonunun gelip gelmediğinin kontrolünü yapmak:

```
JSR READST
AND #64         ;EOF bitini kontrol et (EOF = dosya sonu)
BNE EOF         ;Dosya sonu ise dallan
```

B-23.

Fonksiyon adı : RESTOR
Amaç : Varsayılan sistemi geri yükleyin ve vektörleri kesin
Çağrı adresi : $FF8A (onaltılık) 65418 (onluk)
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, KERNAL ve BASIC rutinlerinde ve kesintilerde kullanılan tüm sistem vektörlerine ilk açılıştaki değerlerini verir (Vektörlerin normal içerikleri için Bellek Haritasına bakınız). KERNAL VECTOR rutini, sistem vektörlerinin herhangi birinin okunması ve değiştirilmesinde kullanılır.

Kullanımı: Bir aşamadan oluşur.
1) Bu rutini çağırın.

Örnek:

```
JSR RESTOR
```

B-24.

Fonksiyon adı : SAVE
Amaç : Belleğin içeriğini bir cihaza saklamak
Çağrı adresi : $FFD8 (onaltılık) 65496 (onluk)
İletişim kayıtları : .A .X, .Y
Hazırlık rutinleri : SETLFS, SETNAM
Hata mesajları : 5, 8, 9, READST
Yığın gereksinimi : Yok
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, belleğin bir bölümünü saklamak etmek için kullanılır. Rutin, akümülatörün gösterdiği sıfırıncı sayfadaki bir dolaylı adresten, .X ve .Y kayıtlarının gösterdiği adrese kadar olan bellek bölgesini, bir g/ç cihazındaki dosyaya saklar. Bu rutinden önce, SETLFS ve SETNAM rutinlerinin çağırılması gerekir. Programı ya da verileri kasette saklamak istiyorsak dosya ismi vermeye gerek yoktur. Fakat başka bir cihazda saklanmak istendiğinde, dosya ismi kullanmamak hataya neden olacaktır.

> **NOT:** Cihaz 0 (klavye), cihaz 2 (RS-232) ve cihaz 3'e (ekran) saklama işlemi yapılamaz. Bu, hataya neden olur ve saklama işlemi durur.

Kullanımı: Beş aşamadan oluşur.

1) SETLFS ve (dosya ismi vermeden kasete kayıt yapmıyorsanız) SETNAM rutinlerini kullanın.

2) Kaydedeceğiniz bellek bölgesinin başlangıç adresini, alt-bayt üst-bayt sırasıyla 0'ıncı sayfadaki iki ardışık adrese yerleştirin.

3) Akümülatöre 0'ıncı sayfadaki iki ardışık adresin birincisini yükleyin.

4) X ve Y kayıtlarına sırasıyla saklanacak bölgenin sonuç adresinin alt ve üst basamak baytlarını yükleyin.

5) Rutini çağırın.

Örnek:

```
LDA #1          ;Cihaz= 1:Teyp
JSR SETLFS
LDA #0          ;Dosya adı yok
JSR SETNAM
LDA PROG        ;Saklama işlemi için başlangıç adresi
STA TXTTAB      ;Alt bayt
LDA PROG+1
STA TXTTAB+1    ;Üst bayt
LDX VARTAB      ;Saklama işlemi için bitiş adresi
LDY VARTAB+1    ;Üst bayt
LDA #<TXTTAB    ;Akümülatöre 0'ıncı sayfadaki ofset'i yükle
JSR SAVE
```

B-25.

Fonksiyon adı : SCNKEY
Amaç : Klavyenin taranması
Çağrı adresi : $FF9F (onaltılık) 65439 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : IOINIT
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, Commodore 64'ün klavyesini tarayarak basılmış tuşları algılar. Kesme işleyicisi rutininin bir parçasıdır. Bir tuşa basıldığında, bu tuşun ASCII değeri klavye deposuna (queue) yerleştirilir. Bu rutin sadece normal IRQ kesmesi devreden çıkarıldığı zaman kullanılmalıdır.

Kullanımı: Bir aşamadan oluşur.

1) Rutini çağırın.

Örnek:

```
GET JSR SCNKEY      ;Klavyeyi tara
    JSR GETIN       ;Karakteri al
    CMP #0          ;Boş karakter mi?
    BEQ GET         ;Evet ise tekrar tara
    JSR CHROUT      ;Yazdır
```

B-26.

Fonksiyon adı : SCREEN
Amaç : Ekran formatını bildirmek
Çağrı adresi : $FFED (onaltılık) 65517 (onluk)
İletişim kayıtları : .X, .Y
Hazırlık rutinleri : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .X, .Y

Tanımlama: Bu rutin, çağırıldıktan sonra ekran formatının özelliklerini bildirir, .X indeksinde kolon sayısı olan 40, . Y indeksinde ise satır sayısı olan 25'in yer alması gibi. Aynı zamanda, programın hangi makinada çalıştırıldığını anlamak için de kullanılabilir. Bu fonksiyon, Commodore 64 programlarınızın başka Commodore cihazlarına uyarlanabilirliğini arttırmak için düşünülmüştür.

Kullanımı: Bir aşamadan oluşur.
1) Rutini çağırın.

Örnek:
```
JSR SCREEN
STX MAXCOL
STX MAXROW
```

B-27.

Fonksiyon adı : SECOND
Amaç : Dinle (LISTEN) için ikincil adresi göndermek
Çağrı adresi : $FF93 (onaltılık) 65427 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : LISTEN
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : 8
Etkilenen kayıtlar : .A

Tanımlama: Bu rutin, bir g/ç cihazına LISTEN ile dinleme emri verildikten sonra o cihaza ikincil adres göndermek amacı ile kullanılır. Konuş (TALK) rutinine ikincil adres göndermek için bu rutin kullanılmaz.

İkincil adres genellikle g/ç işlemleri başlamadan evvel, bir cihaza çeşitli kontrol bilgileri iletmek amacıyla kullanılır.

Seri porta bağlı bir cihaza gönderilecekse, kullanılacak adresin $60'la "OR" işlemine sokulması gerekir.

Kullanımı: İki aşamadan oluşur.
1) Gönderilecek ikincil adresi $60 ile OR'layıp akümülatöre yükleyin.
2) Rutini çağırın.

Örnek:
```
LDA #8
JSR LISTEN
LDA #15
JSR SECOND
```

B-28.

Fonksiyon adı : SETLFS
Amaç : Dosya tanımlamak
Çağrı adresi : $FFBA (onaltılık) 65466 (onluk)
İletişim kayıtları : .A, .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : Yok

Tanımlama: Bu rutin, diğer KERNAL rutinleri için dosya numarası, cihaz adresi ve ikincil adres tanımlar.

Dosya numarası, OPEN rutini ile yaratılmış olan dosya tablosu için, sistem tarafından anahtar olarak kullanılır. Cihaz adresleri 0 ile 31 arasında olabilir. Aşağıdaki kodlar Commodore 64 tarafından CBM cihazları için kullanılır:

Adres	Cihaz
0	Klavye
1	Teyp ünitesi
2	RS-232 cihazı
3	CRT Ekran
4	Seri Porta bağlı yazıcı
8	Seri Porta bağlı disk sürücü

4 ve daha büyük cihaz numaraları, seri porta bağlı cihazlar için kullanılır. Cihaz numarası seri uyarı haberleşim dizisi süresince gönderildikten sonra, gene seri bağlantıda, cihaza ikincil adresi veren bir komut gönderilir. Eğer hiç ikincil adres gönderilmeyecekse Y indeks kaydının 255 ile yüklenmiş olması gerekir.

Kullanımı: Üç aşamadan oluşur.

1) X indeks kaydına dosya numarasını yükleyin.
2) Akümülatöre cihaz numarasını yükleyin.
3) Y indeks kaydına dosya komut yükleyin.

Örnek:

Dosya no.:32, cihaz no.:4, ve komut yok:

```
LDA #32
LDX #4
LDY #255
JSR SETLFS
```

B-29.

Fonksiyon adı : SETMGS
Amaç : Sistem mesajı çıkışlarını kontrol etmek
Çağrı adresi : $FF90 (onaltılık) 65424 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A

Tanımlama: Bu rutin, KERNAL tarafından gönderilen hata ve kontrol mesajlarının basılmasını kontrol eder. Rutin çağırıldığında, akümülatöre bir değer yüklenerek, hata ya da kontrol mesajlarının seçimi sağlanır. FILE NOT FOUND (Dosya bulunmadı); bu bir hata mesajı, PRESS PLAY ON TAPE (teybin play tuşuna basınız) ise bir kontrol mesajıdır.

Bu değerin 6 ve 7'nci bitleri, mesajın nereden geleceğini belirler. 1'nci bit 1 ise, KERNAL'in hata mesajlarından biri, eğer 6'ncı bit 1 ise, kontrol mesajı basılacaktır.

Kullanımı: İki aşamadan oluşur.

1) İstenilen değeri akümülatöre yükleyin.
2) Rutini çağırın.

Örnek:

```
LDA #$40
JSR SETMSG       ;Kontrol mesajları
LDA #$80
JSR SETMSG       ;Hata mesajları
LDA #0
JSR SETMSG       :Tüm KERNAL mesajlarını önle
```

B-30.

Fonksiyon adı : SETNAM
Amaç : Dosya ismi vermek
Çağrı adresi : $FFDB (onaltılık) 65469 (onluk)
İletişim kayıtları : .A, .X, .Y
Hazırlık rutinleri : Yok
Yığın gereksinimi : Yok
Etkilenen kayıtlar : Yok

Tanımlama: Bu rutin, OPEN, SAVE veya LOAD rutinleri için bir dosya ismi konması amacıyla kullanılır. Akümülatöre, dosya isminin uzunluğunun yüklenmesi gerekir. Dosya isminin adresi, .X ve .Y kayıtlarına 6502'nin alt-bayt/üst-bayt formatıyla yüklenmelidir. Adres, dosya ismindeki karakterlerin saklanabileceği herhangi bir bellek adresi olabilir. Dosya ismi istenmiyorsa, dosya isminin uzunluğunun 0 olduğunu göstermek için, akümülatöre 0 yüklenmesi gerekir. Bu durumda, X ve Y kayıtları herhangi bir bellek adresini gösterebilir.

Kullanımı: Dört aşamadan oluşur.

1) Akümülatöre dosya isminin uzunluğunu yükleyin.
2) .X kaydına dosya isminin alt baytını yükleyin.
3) .Y kaydına dosya isminin alt baytını yükleyin.
4) Rutini çağırın.

Örnek:

```
LDA #isim2-isim    ;Dosya isminin uzunluğunu yükle
LDX #<isim         ;Dosya isminin adresini yükle
LDY #>isim
JSR SETNAM
```

B-31.

Fonksiyon adı : SETTIM
Amaç : Sistem saatini kurmak
Çağrı adresi : $FFDB (onaltılık) 65499 (onluk)
İletişim kayıtları : .A, .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : Yok

Tanımlama: Sistem saati bir kesinti rutini ile, saniyenin her 1/60'ında bir anlık (jiffy) düzeltilir. Saat, kendisine 5,184,000 (24 saat)'a kadar sayabilme olanağı verecek şekilde 3 baytlıktır. Bu noktada saat tekrar sıfırlanır. Saatin kurulması için bu rutin çağırılmadan evvel akümülatörün başlangıç zamanının ilk baytını, X kaydının bir sonraki bayt ve Y kaydının da son baytı içermesi gerekir (Anlık durumda).

Kullanımı: Dört aşamadan oluşur.

1) Akümülatöre, saatin kurulması için gereken 3 baytlık sayının ilk baytını yükleyin.
2) X kaydına bir sonraki bayt, yükleyin.
3) Y kaydına bir sonraki bayt, yükleyin.
4) Rutini çağırın.

Örnek:

Saati 1 dakikaya ayarla, 1 dakika=3600 anlık

```
LDA #0          ;En soldaki
LDX #>3600
LDY #<3600      ;En sağdaki
JSR SETTIM
```

B-32.

Fonksiyon adı : SETTMO
Amaç : IEEE bağlantısı zaman aşımı durumunu oluşturur
Çağrı adresi : $FFA2 (onaltılık) 65442 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : Yok

> **NOT:** Bu rutin sadece IEEE ekleme kartı ile kullanır.

Tanımlama: Bu rutin, IEEE bağlantısı için zaman aşımı durumunu 1 yapar. Bayrak bir olduğunda, Commodore 64 IEEE portunda 64 milisaniye bir cihaz bekler. Eğer cihaz bu zaman zarfında Commodore 64'ün geçerli adres veri sinyaline karşılık vermezse, Commodore 64 hata durumu olduğunu anlar ve haberleşme dizisine son verir. Bu rutin, akümülatörün 7'nci biti 0 iken çağırılırsa, zaman aşımı oluşmasına izin verilmiş demektir. Bit 7'nin 1 olması ise zaman aşımı olmasını önler.

NOT: Commodore 64 zaman aşımı kavramını, IEEE kartı kullanan profesyonel bir disk ünitesinde, aranan dosyanın bulunmadığını saptamakta kullanır.

Kullanımı: İki bölüm ve her bölüm iki aşamadan oluşur.

Zaman aşımı bayrağını ayarlamak için:
1) Akümülatörün 7'nci bitini 0 yapın.
2) Rutini çağırın.

Zaman aşımı bayrağını sıfırlamak için:
1) Akümülatörün 7'nci bitini 1 yapın.
2) Rutini çağırın.

Örnek:
Zaman aşımını devre dışı bırakmak:

```
LDA #0
JSR SETTMO
```

B-33.

Fonksiyon adı : STOP
Amaç : STOP tuşunun basılı olup olmadığını kontrol etmek.
Çağrı adresi : $FFE1 (onaltılık) 65505 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : Yok
Etkilenen kayıtlar : .A, .X

Tanımlama: UDTIM rutininin çağırıldığı sırada STOP tuşuna basılırsa, bu çağrıdan dolayı Z bayrağı bir olur ve kanallar varsayılan değerlerini alırlar. Diğer bütün bayraklar değişmeden kalır. STOP tuşuna basılmamışsa, akümülatör klavye taramasının son satırını gösteren baytı içerecektir. Kullanıcı bu yolla, diğer bütün tuşları kontrol edebilir.

Kullanımı: Üç aşamadan oluşur.

1) Bu rutinden önce UDTIM'i çağırın.
2) Rutini çağırın.
3) Sıfır bayrağını kontrol edin.

Örnek:

```
JSR UDTIM       ;STOP için tara
JSR STOP
BNE *+5         ;Tuş basılı değil
JMP READY       ; = ....... STOP
```

B-34.

Fonksiyon adı : TALK
Amaç : Seri bağlantıdaki cihaza konuş (TALK) komutu vermek
Çağrı adresi : $FFB4 (onaltılık) 65460 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : Yok
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : 8
Etkilenen kayıtlar : .A

Tanımlama: Bu rutini kullanmak için, akümülatöre 0 ile 31 arasında bir cihaz no.'su yüklemek gerekir. Çağrıldığı zaman ise bu rutin, cihaz no.'sunu (bit, bit OR (VEYA)'layarak) konuşma adresine çevirir. Daha sonra veri, bir komut gibi seri bağlantıyı kullanarak iletilir.

Kullanımı: İki aşamadan oluşur.

1) Cihaz no.'sunu akümülatöre yükleyin.
2) Rutini çağırın.

Örnek:

4 No.'lu cihaza konuşmasını emret:

```
LDA #4
JSR TALK
```

B-35.

Fonksiyon adı : TKSA
Amaç : Konuşmadaki (TALK) cihaza ikincil adres göndermek
Çağrı adresi : $FF96 (onaltılık) 65430 (onluk)
İletişim kayıtları : .A
Hazırlık rutinleri : TALK
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : 8
Etkilenen kayıtlar : .A

Tanımlama: Bu rutin, bir TALK (konuşma) cihazı için, seri bağlantıya ikincil adres gönderir. Rutin, akümülatörde yer alan 0 ile 31 arasındaki sayı ile çağırılmalıdır. Rutin bu sayıyı seri bağlantıya ikincil adres komutu olarak gönderir. Yalnızca TALK rutininden sonra çağırılabilir. LISTEN (dinle) rutininden sonra çalışmayacaktır.

Kullanımı: Üç aşamadan oluşur.

1) TALK rutinini kullanın.
2) Akümülatöre ikincil adresi yükleyin.
3) Rutini çağırın.

Örnek:

4 No.'lu cihaza 7 no.'lu komut ile konuşmasını emret:

```
LDA #4
JSR TALK
LDA #7
JSR TKSA
```

B-36.

Fonksiyon adı : UDTIM
Amaç : Sistem saatini tanımlama (update)
Çağrı adresi : $FFEA (onaltılık) 65514 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A, .X

Tanımlama: Bu rutin, sistem saatini yeniden günceller (update). Normal olarak. Bu rutin KERNAL kesinti (interrupt) rutini tarafından saniyenin her 60'da birinde çağırılır. Eğer kullanıcı programları, kendi kesintilerini işliyorlarsa, zamanı düzeltmek için bu rutinin çağırılması gerekir. Buna ek olarak, STOP tuşu işlevsel olarak kaldığı takdirde, STOP tuşu rutininin çağırılması gerekir.

Kullanımı: Bir aşamadan oluşur.

1) Rutini çağırın.

Örnek:

```
JSR UDTIM
```

B-37.

Fonksiyon adı : UNLSN
Amaç : Cihazlara dinlememe (UNLISTEN) komutu göndermek
Çağrı adresi : $FFAE (onaltılık) 65454 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : Yok
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : 8
Etkilenen kayıtlar : .A

Tanımlama: Bu rutin, seri bağlantıdaki tüm cihazlara, Commodore 64'ten gönderilen verileri almamalarını emreder. Rutini çağırarak, seri bağlantıdan konuşmama (UNLISTEN) komutunun gönderilmesi sağlanır, fakat sadece daha önce dinlemeleri için komut verilmiş olan cihazlar etkilenir. Normal olarak. Commodore 64'ün diğer cihazlara veri göndermesi bittikten sonra çağrılır. Dinleyen cihazlara, seri bağlantıdan uzaklaşması için gönderilen konuşmama (UNLISTEN) komutları, başka amaçlar için de kullanılır.

Kullanımı: Bir aşamadan oluşur.

1) Rutini çağırın.

Örnek:

```
JSR UDTIM
```

B-38.

Fonksiyon adı : UNTLK
Amaç : Cihazlara konuşmama (UNTALK) komutu göndermek
Çağrı adresi : $FFAB (onaltılık) 65451 (onluk)
İletişim kayıtları : Yok
Hazırlık rutinleri : Yok
Hata mesajları : READST'ye bakınız
Yığın gereksinimi : 8
Etkilenen kayıtlar : .A

Tanımlama: Bu rutin, seri bağlantıdan konuşmama (UNTALK) komutunun gönderilmesini sağlar. Bu komut iletildiğinde, daha önce konuş (TALK) komutu almış tüm cihazlar, veri göndermeyi durdururlar.

Kullanımı: Bir aşamadan oluşur.

1) Rutini çağırın.

Örnek:

JSR UNTALK

B-39.

Fonksiyon adı : VECTOR
Amaç : RAM vektörlerinin yönetimi
Çağrı adresi : $FF8D (onaltılık) 65421 (onluk)
İletişim kayıtları : .X, .Y
Hazırlık rutinleri : Yok
Hata mesajları : Yok
Yığın gereksinimi : 2
Etkilenen kayıtlar : .A, .X, .Y

Tanımlama: Bu rutin, RAM'de bulunan tüm sistem vektörlerinin sıçrama adreslerinin yönetilmesini sağlar. Akümülatörün taşıma biti 1 iken bu rutin çağırıldığında, RAM vektörlerinin o andaki içerikleri, .X ve .Y kayıtları tarafından gösterilen listeye kaydedilir. Taşıma biti 0 iken çağırıldığında ise .X ve .Y kayıtlarının gösterdiği kullanıcı listesi, sistem RAM vektörlerine transfer edilir. RAM vektörleri bellek haritasında liste halinde gösterilmiştir.

NOT: Bu rutinin kullanırken çok dikkatli olmak gerekir. Bu rutini kullanmanın en iyi yolu, önce tüm vektörlerin içeriklerini kullanıcı-alanına okumak, istediklerinizi değiştirmek ve daha sonra içerikleri sistem vektörlerine kopyalamaktır.

Kullanımı: İki bölüm ve her bölüm üç aşamadan oluşur.

Sistem ram vektörlerinin okunması:

1) Taşımayı ayarlayın.
2) .X ve .Y kayıtlarına, vektörlerin yerleştirileceği adresleri ayarlayın.
3) Rutini çağırın.

Sistem ram vektörlerinin yüklenmesi:

1) Taşımayı sıfırlayın.

2) RAM'den yüklenmesi gereken vektör listesinin adresini, .X ve .Y kayıtlarına yükleyin.

3) Rutini çağırın.

Örnek:

Yeni sistem için giriş rutinlerini değiştir:

```
LDX #<kullanıcı
LDY #>kullanıcı
SEC
JSR VECTOR          ;Eski vektörleri oku
LDA #<yeni-giriş    ;Girişleri değiştir
STA kullanıcı+10
LDA #>yenigiriş
STA kullanıcı+11
LDX #<kullanıcı
LDY #>kullanıcı
CLC
JSR VECTOR          ;Sistemi değiştir
...
kullanıcı *=*+26
```

HATA KODLARI

Aşağıda, KERNAL rutinlerini kullanırken oluşabilecek hata uyarılarının listesi verilmiştir. KERNAL rutini sırasında bir hata oluşursa, akümülatörün taşıma biti ayarlanır ve hata numarası akümülatöre kaydedilir.

> **NOT:** Bazı KERNAL giriş/çıkış rutinleri hata uyarıları için bu kodları kullanmazlar. Bunun yerine, hatalar KERNAL READST rutini tarafından tanımlanır.

NUMARA	ANLAMI
0	Rutin STOP tuşu ile durdurulmuştur.
1	Çok fazla dosya açık
2	Dosya daha önce açılmış
3	Dosya açılmamış
4	Dosya bulunamadı
5	Cihaz hazır değil
6	Giriş dosyası değil
7	Çıkış dosyası değil
8	Dosya ismi yok
9	Cihaz numarası kuraldışı
240	RS-232 tamponunun yerleştirilmesi/kaldırılması
	Bellek-başlangıcı değişikliği

BASIC'TEN MAKİNE DİLİ KULLANIMI

Commodore 64'te makine dilini ve BASIC'ı birlikte kullanmanın, yeni komutlar üretmenin ve ROM'daki belli bölümleri kullanmanın çeşitli yöntemleri vardır. Bunların başlıca 5 tanesi şunlardır:

1) BASIC SYS yönergesi
2) BASIC USR fonksiyonu
3) RAM giriş/çıkış vektörlerinden birini değiştirmek
4) RAM kesinti vektörlerinden birini değiştirmek
5) CHRGET rutinini değiştirmek

1) BASIC yönergesi SYS X, X adresinde bulunan makine dili bir programa JSR ile dallanıp bu programın işletilmesini sağlar. Rutinin RTS komutuyla bitmesi gerekir. Kontrol tekrar BASIC'e geçer.

Değişkenler ve bunların makine dilindeki karşılıklarına çevrilmesi, BASIC'ten makine dili rutinlere geçirilmesi, BASIC PEEK ve POKE komutları yardımıyla yapılır.

Makine dili ile BASIC'i birleştirmenin en faydalı yöntemi SYS komutudur. PEEK ve POKE, birden fazla değişkenin kolaylıkla transfer edilmesini sağlarlar. Bir programın içinde birden fazla SYS komutu olabilir ve bunların her biri aynı ya da değişik bir makine dili rutini için olabilir.

2) BASIC'teki USR(X) fonksiyonu, kontrolü, başlangıç adresinin alt ve üst baytlar 785 ve 786 adreslerinde bulunan makine dili programına devreder. (Adresler, standart alt bayt-üst bayt formatında tutulur). X değeri bulunur ve $61 adresinden başlayarak yerleştirilmiş olan 1 numaralı ondalık sayı akümülatörünü kullanarak, makine dili rutinine aktarılır. Rutin işini bitirince, eğer geriye bir sonuç göndermek gerekiyorsa, bir değer, ondalık sayı akümülatörü kullanılarak tekrar BASIC programına aktarılabilir. BASIC'e geri dönülmesi için, makine dilindeki rutin, RTS komutuyla bitmek zorundadır.

Dolaylı vektör oluşturmanız gerektiğinden, bu yönerge SYS'den farklıdır. Ayrıca değişkenin formatı da farklıdır (ondalık sayı formatı). Eğer birden fazla makine dili rutini kullanılıyorsa, dolaylı vektörün değiştirilmesi gerekir.

3) Sayfa 3'e yerleştirilmiş olan vektör tablosundan erişilen (adresleme çeşitlerindeki sıfırıncı sayfaya bakınız), giriş/çıkış ya da BASIC içindeki rutinlerin herhangi birisi, kullanıcı kodu ile değiştirilebilir ya da işlenebilir. 2-Baytlık vektörlerin her biri, işletim sistemi tarafından kullanılan alt-bayt ve üst-bayt adresten oluşmuştur.

Vektörleri değiştirmek için KERNAL VEKTOR rutinini kullanmak en güvenilir yoldur. Fakat tek bir vektörü değiştirmek için POKE komutunu da kullanabilirsiniz. Yeni bir vektör, standart sistem rutininin yerini değiştirmeye ya da çoğaltmaya yarayacak, kullanıcı tarafından hazırlanmış rutini gösterecektir. Uygun BASIC komutu işlendiğinde, kullanıcı rutini de işlenecektir. Eğer kullanıcı rutini işlendikten sonra normal sistem rutininin de işlenmesi gerekiyorsa, kullanıcı programı, önceden vektörün içinde bulunan adrese dallanmalıdır (JMP ile). Eğer gerekmiyorsa, kontrolün BASIC'e devredilmesi için rutin, RTS ile bitmelidir. Örneğin, normalde her 1/60 saniyede devreye giren interrupt rutini $EA31'de başlar ve bu vektör $0314-$0315 adreslerinde bulunur. Eğer biz $C000'da yeni

bir rutin yazarsak, bu vektörleri $00 ve $C0 olarak değiştirmeliyiz ve kendi yazdığımız rutinin en son komutu da JMP $EA31 olmalıdır. Böylece hem kendi programımız hem de orijinal rutin çalışmış olur. Ancak bu vektörleri BASIC ile değiştirmek, yani POKE ile değiştirmek saniyenin 1/60'ından uzun süreceği için, bilgisayar kilitlenecektir. Bu nedenle makine dilinde:

```
SEI
LDA #$00
STA $0314
LDA #$C0
STA $0315
CLI
RTS
```

4) SİSTEM KESİNTİ VEKTÖR'ü (IRQ) değiştirilebilir. Her 1/60 saniyede, işletim sistemi kontrolü bu vektör tarafından belirlenen rutine devreder. KERNAL bunu genelde zamanlama, klavye tarama, vs. için kullanır. Eğer bu teknik kullanılıyorsa ve sizin yazdığınız rutin CIA çipinin gereksinim duyduğu düzeltmeleri yapmıyorsa işlem bittiğinde kontrolü, JMP $EA31 ile sistem rutinine devretmelisiniz. (CIA bu rutin ile işleniyorsa rutinin sonunda RTI (kesintiden geri dön) ile bitirilmesi gerekir).

Bu metot, bir BASIC programı ile aynı anda olması gereken görevler için oldukça faydalıdır, fakat oldukça zordur.

NOT: Bu vektörü değiştirmeden önce daima kesintileri devre dışı bırakın! (SEI komutu ile).

5) CHRGET rutini, BASIC tarafından her bir karakteri/belirteci almak zorundadır. Bu, yeni BASIC komutlarının eklenmesini kolaylaştırır. Genel olarak her yeni komut, kullanıcı tarafından yazılmış makine dili bir rutin ile işlenmelidir. Bu metodu kullanmanın en basit yolu, yeni komutların önlerinde bulunacak yeni bir karakter (örneğin: @) tanımlamaktır. Yeni CHRGET rutini, özel karakteri arayacaktır. Eğer yoksa, kontrol normal BASIC CHRGET rutinine devredilecektir. Eğer özel karakter mevcut ise, yeni komut yorumlanır ve makine dili program tarafından işlenir. Bu, ilave komutların aranması için gereken, fazladan işletim zamanı harcanmasını önleyecektir. Bu tekniğe genellikle kama “wedge” denir.

MAKİNE DİLİ RUTİNLERİ NEREYE KONULUR?

Commodore64'te rutinlerin 4K bayttan küçük olması şartıyla makine dili rutinlerinin yerleştirileceği en uygun yer $C000-$CFFF arasıdır. Belleğin bu kısmı BASIC tarafından kullanılmayacağı için, rutinleriniz hasar görmeyecektir.

Eğer herhangi bir nedenden ötürü, makine dili rutinlerin $C000'a yerleştirilmesi mümkün değilse, (örneğin, rutinin 4K bayttan büyük olması gibi) BASIC'in bellek alanının sonundan itibaren bir bölümün, bu iş için ayrılması gerekir. Bellek normal olarak $9FFF'de biter. Bellek tavanı, KERNAL rutini MEMTOP ya da aşağıdaki BASIC yönergesi kullanılarak değiştirilebilir.

```
10 POKE51,L:POKE52,H:POKE55,L:POKE56,H:C
LR
```

Buradaki H ve L, belleğin yeni tavanının alt ve üst baytlarıdır. Örneğin $9000'den $9FFF'e kadar olan alanın makine dili için ayrılmasını istiyorsak aşağıdaki satırı yazmalıyız:

```
10 POKE51,0:POKE52,144:POKE55,0:POKE56,1
44:CLR
```

MAKİNE DİLİ NASIL GİRİLİR?

Makine dili programları bir BASIC programına eklemek için 3 metot kullanılır. Bunlar:

1) DATA YÖNERGELERİ:

READ komutu kullanarak DATA yönergesindeki verileri okumak ve bu verileri POKE komutu ile programın başlangıcındaki belleğe yerleştirerek makine dili rutinler ekleyebiliriz. Bu en kolay yöntemdir. Programın iki bölümünü de saklamak için özel metotlar gerekmez ve hataların düzeltilmesi de oldukça kolaydır. Tek dezavantajı da fazla bellek harcanması ve POKE işlemi sırasında bekletmesidir. Bu yüzden, bu metot sadece küçük rutinler için kullanışlıdır.

```
10 RESTORE:FORX=1TO9:READA:POKE12*4096+X
,A:NEXT
20 REM GERI
30 REM KALAN
40 REM BASIC
50 REM PROGRAM SATIRLARI
1000 DATA 161,1,204,204,204,204,204,204,
96
```

2) MAKİNE DİLİ DENETLEYİCİSİ (64 MON):

Bu program, ONALTILI ya da SEMBOLİK kodlarda program yazmanızı ve programın yer alacağı bellek miktarından tasarruf etmenizi sağlar. Bu metodun avantajları: Makine dili programların kolayca girilebilmesi, hata yakalama ve düzeltmede yardımcı olması, yükleme ve saklama işlemleri için oldukça hızlı çalışmasıdır. Dezavantajları ise, başlangıçta teyp ya da diskten yüklemek için bir BASIC programa gerek duyulmasıdır. (Daha fazla ayrıntı için makine dili bölümündeki 64MON'a bakın).

ÖRNEK:

Aşağıdaki örnek 64MON ile hazırlanmış olan makina dili bir rutin kullanan bir BASIC programıdır ve teyp kullanılmıştır.

```
10 IF FLAG=1 THEN 20
15 FLAG=1:LOAD "MAKINE DILI RUTIN ADI",1
,1
20 REM KALAN BASIC PROGRAM SATIRLARI
```

3) EDİTÖR/ASSEMBLER PAKETİ:

Makine dili monitörü ile aynı avantajlara sahiptir, fakat programların girilmesi daha kolaydır. Dezavantajları da makine dili monitörünün dezavantajlarıyla aynıdır.

COMMODORE 64 BELLEK HARİTASI:

(Aşağıda “bayrak” olarak belirtilen hafıza birimleri İngilizce’de FLAG olarak belirtilen işlemlerde kullanılırlar.)

İSİM	ADRES		TANIMLAMA
	ONALTILIK	ONLUK	
D6510	0000	0	6510 üzerindeki veri yön kaydı
R6510	0001	1	6510 üzerindeki 8-bit giriş/çıkış kaydı
	0002	2	Kullanılmıyor
ADRAY1	0003-0004	3-4	Sıçrama (Jump) vektörü: ondalık-tamsayı dönüşümü
ADRAY2	0005-0006	5-6	Sıçrama (Jump) vektörü: tamsayı-ondalık dönüşümü
CHARAC	0007	7	Karakter arama
ENDCHR	0008	8	Bayrak: yazınsal dizinin sonunu belirten tırnağın aranması
TRMPOS	0009	9	Ekran kolonunun son TAB pozisyonundan uzaklığı
VERCK	000A	10	Bayrak: 0 = yükleme, 1 = doğrulama
COUNT	000B	11	Giriş tamponu göstergeci/indis sayısı
DIMFLG	000C	12	Bayrak: varsayılan dizi boyutu
VALTYP	000D	13	Veri tipi: $FF = yazınsal dizi, $00 = sayısal
INTFLG	000E	14	Veri tipi: $80 = tamsayı, $00 = ondalık sayı
GARBFL	000F	15	Bayrak: DATA tarama/LIST için tırnak işareti /hafıza boşaltma rutini
SUBFLG	0010	16	Bayrak: indis referansı/kullanıcı fonksiyonu çağrısı
INPFLG	0011	17	Bayrak: $00 = INPUT (giriş), $40=GET (al). $98 = READ (oku)
TANSGN	0012	18	Bayrak: TAN işareti/karşılaştırma sonucu
	0013	19	Bayrak: INPUT uyarısı
LINNUM	0014-0015	20-21	Geçici: tamsayı değeri
TEMPPT	0016	22	Göstergeç: geçici yazınsal dizi yığını
LASTPT	0017-0018	23-24	Son geçici yazınsal dizi adresi
TEMPST	0019-0021	25-33	Geçici yazınsal diziler için yığın (STACK)
INDEX	0022-0025	34-37	Yardımcı göstergeç alanı

İSİM	ADRES		TANIMLAMA
	ONALTILIK	ONLUK	
RESHO	0026-002A	38-42	Çarpma işleminin ondalık-sayı sonucu
TXTTAB	002B-002C	43-44	Göstergeç: BASIC Metninin başlangıcı
VARTAB	002D-002E	45-46	Göstergeç: BASIC değişkenlerinin başlangıcı
ARYTAB	002F-0030	47-48	Göstergeç: BASIC dizilerinin başlangıcı
STREND	0031-0032	49-50	Göstergeç: BASIC dizilerinin sonu (+ 1)
FRETOP	0033-0034	51-52	Göstergeç: Dizgi saklanımının sonu
FRESPC	0035-0036	53-54	Yardımcı Dizgi Göstergeci
MEMSIZ	0037-0038	55-56	Göstergeç: BASIC'ın kullanıldığı en yüksek adres
CURLIN	0039-003A	57-58	Bulunduğunuz BASIC satır numarası
OLDLIN	003B-003C	59-60	Bir evvelki BASIC satır numarası
OLDTXT	003D-003E	61-62	Göstergeç: CONT için BASIC yönergesi
DATLIN	003F-0040	63-64	O andaki DATA satırının no.'su
DATPTR	0041-0042	65-66	Göstergeç: Bulunduğunuz DATA adresi
INPPTR	0043-0044	67-68	Vektör: INPUT rutini
VARNAM	0045-0046	69-70	O andaki BASIC değişkeninin ismi
VARPNT	0047-0048	71-72	Göstergeç: O andaki BASIC değişkeni verisi
FORPNT	0049-004A	73-74	Göstergeç: FOR/NEXT için indis değişkeni
	004B-0060	75-96	Geçici göstergeç/veri alanı
FACEXP	0061	97	Ondalık sayı akümülatörü 1: Üs
FACHO	0062-0065	98-101	Ondalık sayı akümülatörü 1: Taban
FACSGN	0066	102	Ondalık sayı akümülatörü 1: İşaret
SGNFLG	0067	103	Göstergeç: Serileri değerlendirme sabiti
BITS	0068	104	Ondalık akümülatör 1: Taşma hanesi
ARGEXP	0069	105	Ondalık akümülatör 1: Üs
ARGHO	006A-006D	106-109	Ondalık akümülatör 1: Taban
ARGSGN	006E	110	Ondalık akümülatör 1: İşaret
AAISGN	006F	111	İşaretlerin karşılaştırılmasının sonucu: akümülatör 1 ve akümülatör 2
FACOV	0070	112	Ondalık aküm. 1: Alt basamak (yuvarlayarak)
FBUFPT	0071-0072	113-114	Göstergeç: Teyp tampon
CHRGET	0073-008A	115-138	Altprogram: BASIC metninden bir sonra gelen baytın alınması
CHRGOT	0079	121	Metindeki aynı baytın tekrar alınması
TXTPTR	007A-007B	122-123	Göstergeç: BASIC metninin o andaki baytı
RNDX	008B-008F	139-143	Ondalıklı RND fonksiyonun değeri
STATUS	0090	144	Kernal G/Ç Statü sözcüğü: ST
STKEY	0091	145	Bayrak: STOP tuşu/RVS tuşu
SVXT	0092	146	Teyp için zamanlama sabiti
VERCK	0093	147	Bayrak: 0=yükleme, 1=doğrulama
C3PO	0094	148	Bayrak: Seri bağlantı-tamponlanmış çıkış karakteri
BSOUR	0095	149	Seri bağlantı için tamponlanmış karakter
SYNO	0096	150	Kaset eşzaman sayısı

İSİM	ADRES		TANIMLAMA
	ONALTILIK	ONLUK	
	0097	151	Geçici veri alanı
LDTND	0098	152	Açık dosya sayısı/dosya tablosu için indeks
DFLTN	0099	153	Varsayılan giriş cihazı (0)
DFLTO	009A	154	Varsayılan çıkış (CMD) cihazı (3)
PRTY	009B	155	Teyp karakter eşliği (parity)
DPSW	009C	156	Bayrak: Kasetten bayt alındı
MSGFLG	009D	157	Bayrak: $80=doğrudan mod, $00=program
PTR1	009E	158	Teyp geçişi 1 hata günlüğü
PTR2	009F	159	Teyp geçişi 2 hata günlüğü
TIME	00A0-00A2	160-162	Gerçek zaman anlık saat (yaklaşık) 1/60 saniye
	00A3-00A4	163-164	Geçici data alanı
CNTON	00A5	165	Eşzamanlı kaset geri sayımı
BUFPNT	00A6	166	Göstergeç: Teyp giriş/çıkış tamponu
INBIT	00A7	167	RS-232 giriş bitleri/kaset için geçici bölge
BITCI	00A8	168	RS-232 giriş biti sayımı/kaset için geçici bölge
RINONE	00A9	169	RS-232 bayrak: Başlangıç biti kontrolü
RIDATA	00AA	170	RS-232 giriş bayt tamponu/kaset için geçici böl.
RIPRTY	00AB	171	RS-232 giriş eşliği/kaset kısa sayımı
SAL	00AC-00AD	172-173	Gösterge: Teyp tamponu/ekran kayması
EAL	00AE-00AF	174-175	Teyp sonu adresi/program sonu
CMPO	00B0-00B1	176-177	Teyp zamanlama sabitleri
TAPE1	00B2-00B3	178-179	Göstergeç: Teyp tamponunun başlangıcı
BITTS	00B4	180	RS-232 çıkış biti sayımı/kaset için geçici bölge
NXTBIT	00B5	181	RS-232'de bir sonra gönderilecek bit/teyp EOF bayrağı
RODATA	00B6	182	RS-232 çıkış bayt tamponu
FNLEN	00B7	183	İşlemdeki dosya ismi uzunluğu
LA	00B8	184	İşlemdeki dosya numarası
SA	00B9	185	İşlemdeki ikincil adres
FA	00BA	186	İşlemdeki cihaz numarası
FNADR	00BB-00BC	187-188	Göstergeç: İşlemdeki dosya ismi
ROPRTY	00BD	189	RS-232 çıkış eşliği/kaset için geçici bölge
FSBLK	00BE	190	Kaset okuma/yazma bloğu sayımı
MYCH	00BF	191	Seri sözcük tamponu
CAS1	00C0	192	Teyp motoru kilitlemesi
STAL	00C1-00C2	193-194	G/Ç başlangıç adresi
MEMUSS	00C3-00C4	195-196	Teyp yüklemesi için geçici bölge
LSTX	00C5	197	Basılı tuş: CHR$(n) 0=tuşa basılmamış
NDX	00C6	198	Klavye tamponunda bulunan karakter sayısı
RVS	00C7	199	1=evet, 0=hayır (Negatif gösterim bayrağı)
INDX	00C8	200	Göstergeç: INPUT için mantıksal satır sonu
LXSP	00C9-00CA	201-202	INPUT başlangıcında takipçinin X-Y konumu
SFDX	00CB	203	Bayrak: SHIFT ile karakter yazımı
BLNSW	00CC	204	Takipçi yanıp sönebilir. 0=parlayan takipçi

İSİM	ADRES		TANIMLAMA
	ONALTILIK	ONLUK	
BLNCT	00CD	205	Zamanlayıcı: Takipçiyi yakıp söndürmek için geri sayım
GOBLN	00CE	206	Takipçinin üzerinde bulunduğu karakter
BLNON	00CF	207	Bayrak: Takipçinin son yanıp sönme hali (on/off)
CRSW	00D0	208	Bayrak: Klavyeden INPUT veya GET
PNT	00D1-00D2	209-210	Göstergeç: İşlemdeki ekran satırı
PNTR	00D3	211	İşlemdeki satırda takipçinin bulunduğu sütun
QTSW	00D4	212	Bayrak: Tırnak modunda editör, $00= hayır
LNMX	00D5	213	Ekran satırının fiziksel uzunluğu
TBLX	00D6	214	Takipçinin olduğu satırın fiziksel satır numarası
	00D7	215	Geçici veri alanı
INSRT	00D8	216	Bayrak: Araya karakter yerleştirme modu, >0=araya yerleştirilen karakter sayısı
LDTB1	00D9-00F2	217-242	Ekran satırı bağlantı tablosu/editör için geçici bölge
USER	00F3-00F4	243-244	Göstergeç: İşlemdeki ekran rengi RAM yeri
KEYTAB	00F5-00F6	245-246	Vektör: Klavye çözüm tablosu
RIBUF	00F7-00F8	247-248	RS-232 giriş tampon göstergeci
ROBUF	00F9-00FA	249-250	RS-232 çıkış tampon göstergeci
FREKZP	00FB-00FE	251-254	Kullanıcı programları için boş 0'ıncı sayfa alanı
BASZPT	00FF	255	BASIC geçici veri alanı
	0100-01FF	256-511	Mikroişlemci sistem yığın alanı
	0100-010A	256-266	Ondalıktan dizgiye çalışma alanı
BAD	0100-013E	256-318	Teyp girişi hata kayıdı
BUF	0200-0258	512-600	Sistem giriş tamponu
LAT	0259-0262	601-610	KEANAL tablosu: Aktif mantıksal dosya sayısı
FAT	0263-026C	611-620	KERNAL tablosu: Her dosya için cihaz sayısı
SAT	026D-0276	621-630	KEANAL tablosu: Her dosya için ikincil adres
KEYD	0277-0280	631-640	Klavye tampon kuyruğu (FIFO)
MEMSTR	0281-0282	641-642	Göstergeç: BASIC için belleğin taban adresi
MEMSIZ	0283-0284	643-644	Göstergeç: BASIC için bellek tavan adresi
TIMOUT	0285	645	Bayrak: IEEE zaman aşımı için KERNAL değişkeni
COLOR	0286	646	İşlemdeki karakter renk kodu
GDCOL	0287	647	Takipçinin üzerinde bulunduğu fon rengi
HIBASE	0288	648	Ekran belleğinin tavanı (Sayfa)
XMAX	0289	649	Klavye tamponunun büyüklüğü
RPTFLG	028A	650	Bayrak: Tuşlar için tekrarlama fonksiyonu, $80=tekrarla
KOUNT	028B	651	Hız sayacı tekrarlama
DELAY	028C	652	Tekrarlamayı geciktirme hız sayacı
SHFLAG	028D	653	Bayrak: Klavye SHIFT tuşu/CTRL tuşu/C= tuşu
LSTSHF	028E	654	Son, klavye SHIFT görüntüsü
KEYLOG	028F-0290	655-656	Vektör: Klavye tablosu durumu

İSİM	ADRES		TANIMLAMA
	ONALTILIK	ONLUK	
MODE	0291	657	Bayrak: $00=SHIFT tuşunun işlevini durdurmak, $80=SHIFT tuşunu işleme sokmak
AUTODN	0292	658	Bayrak: Otomatik aşağı kayma, 0=ON
M51CTR	0293	659	RS-232: Kontrol kaydı görüntüsü
M51CDR	0294	660	RS-232: Komut kaydı görüntüsü
M51AJB	0295-296	661-662	RS-232 standart olmayan BPS (Zaman/2-100) USA
RSSTAT	0297	663	RS-232: 6551 durum kaydı görüntüsü
BITNUM	0298	664	RS-232 gönderilecek bit sayısı
BAUDOF	0299-029A	665-666	RS-232 baud hızı: Tam bit süresi (μs)
RIDBE	029B	667	Giriş tamponunun sonunu gösteren RS-232 indeksi
RIDBS	029C	668	RS-232 giriş tamponu başlangıcı (sayfa)
RODBS	029D	669	RS-232 çıkış tamponu başlangıcı (sayfa)
RODBE	029E	670	Çıkış tamponu sonuna kadar RS-232 indeksi
IRQTMP	029F-02A0	671-672	Teyp giriş/çıkış süresince IRQ vektörünü saklama yeri
ENABL	02A1	673	RS-232 çalışır durumda
	02A2	674	Kaset giriş/çıkış'ı sırasındaTOD duyumu
	02A3	675	Kaset okuması için geçici saklanım yeri
	02A4	676	Kaset okuması için geçici DR1 IRQ göstergeci
	02A5	677	Satır indeksi geçici bölge
	02A6	678	PAL/NTSC bayrağı, 0=NTSC, 1=PAL
	02A7-02FF	679-767	Kullanılmıyor
IERROR	0300-0301	768-769	Vektör: BASIC hata uyarısı yazılır
IMAIN	0302-0303	770-771	Vektör: BASIC sıcak başlangıç
ICRNCH	0304-0305	772-773	Vektör: BASIC metnini tokenize etme
IQPLOP	0306-0307	774-775	Vektör: BASIC metninin listesi
IGONE	0308-0309	776-777	Vektör: BASIC karakter gönderimi
IEVAL	030A-030B	778-779	Vektör: BASIC simge değerlendirmesi
SAREG	030C	780	6502 A kaydı saklanımı
SXREG	030D	781	6502 X kaydı saklanımı
SYREG	030E	782	6502 Y kaydı saklanımı
SPREG	030F	783	6502 SP kaydı saklanımı
USRPOK	0310	784	USR işlevi JUMP komutu ($4C)
USRADD	0311-0312	785-786	USR adresi alt-bayt/üst-bayt
	0313	787	Kullanılmıyor
CINV	0314-0315	788-789	Vektör: Donanım IRQ yarıda kesme
CBINV	0316-0317	790-791	Vektör: BRK komutu yarıda kesme
NMINV	0318-0319	792-793	Vektör: Engellenemez yarıda kesme
IOPEN	031A-031B	794-795	KERNAL OPEN rutini vektörü
ICLOSE	031C-031D	796-797	KERNAL CLOSE rutini vektörü
ICHKIN	031E-031F	798-799	KEANAL CHKIN rutini vektöru
ICKOUT	0320-0321	800-801	KEANAL CHKOUT rutini vektörü

İSİM	ADRES		TANIMLAMA
	ONALTILIK	ONLUK	
ICLRCH	0322-0323	802-803	KERNAL CLRCHN rutini vektörü
IBASIN	0324-0325	804-805	KERNAL CHRIN rutini vektörü
IBSOUT	0326-0327	806-807	KERNAL CHROUT rutini vektörü
ISTOP	0328-0329	808-809	KERNAL STOP rutini vektörü
IGETIN	032A-032B	810-811	KERNAL GETIN rutini vektörü
ICLALL	032C-032D	812-813	KERNAL CLALL rutini vektörü
USRCMD	032E-032F	814-815	Kullanıcı tanımlı vektör
ILOAD	0330-0331	816-817	KERNAL LOAD rutini vektörü
ISAVE	0332-0333	818-819	KERNAL SAVE rutini vektörü
	0334-033B	820-827	Kullanılmıyor
TBUFFA	033C-03FB	828-1019	Teyp giriş/çıkış tamponu
	03FC-03FF	1020-1023	Kullanılmıyor
VICSCN	0400-07FF	1024-2047	1024 bayt ekran bellek alanı
	0400-07E7	1024-2023	Video matrisi:25 satır x 40 kolon
	07F8-07FF	2040-2047	Yaratık veri göstergeçleri
	0800-9FFF	2048-40959	Normal BASIC programlama alanı
	8000-9FFF	32768-40959	VSP kartuş ROM-8192 bayt
	A000-BFFF	40960-49151	BASIC ROM-8192 bayt (veya 8K RAM)
	C000-CFFF	49152-53247	RAM-4096 bayt
	D000-DFFF	53248-57343	Giriş/Çıkış cihazları ve renk RAM'ı veya karakter üretici ROM veya RAM-4096 bayt
	E000-FFFF	57344-65535	KERNAL ROM-8192 bayt (veya 8K RAM)

COMMODORE 64 GİRİŞ/ÇIKIŞ ATAMALARI

ADRES		BİT	TANIMLAMA
ONALTILIK	ONLUK		
0000	0	7-0	MOS 6510 veri yön kaydı (XX101111) bit=1: çıkış, bit=0: giriş, X=1 veya 0 olabilir, önemsiz
0001	1		MOS 6510 mikroişlemci üzerindeki giriş/çıkış portu
		0	/LORAM sinyali (0=BASIC ROM’u devreden çıkartır)
		1	/HIRAM sinyali (0=Kernal ROM’u devreden çıkartır)
		2	/CHAREN sinyali (0=Karakter ROM'u devreye sokar)
		3	Teyp veri çıkış hattı
		4	Teyp açma kapama duyumu. 1=anahtar kapalı
		5	Teyp motor kontrolü. 0=Açık, 1=Kapalı
		6-7	Tanımlanmamış
D000-D02E	53248-54271		MOS 6566 video arabirim kontrolörü (VIC)
D000	53248		0’ıncı yaratığın X konumu
D001	53249		0’ıncı yaratığın Y konumu

ADRES		BİT	TANIMLAMA
ONALTILIK	ONLUK		
D002	53250	7-0	1'inci yaratığın X konumu
D003	53251		1'inci yaratığın Y konumu
D004	53252		2'nci yaratığın X konumu
D005	53253		2'nci yaratığın Y konumu
D006	53254		3'üncü yaratığın X konumu
D007	53255		3'üncü yaratığın Y konumu
D008	53256		4'üncü yaratığın X konumu
D009	53257		4'üncü yaratığın Y konumu
D00A	53258		5'inci yaratığın X konumu
D00B	53259		5'inci yaratığın Y konumu
D00C	53260		6'ncı yaratığın X konumu
D00D	53261		6'nci yaratığın Y konumu
D00E	53262		7'nci yaratığın X konumu
D00F	53263		7'nci yaratığın Y konumu
D010	53264		0-7'nci yaratıkların ESB X konumu (X koord. en soldaki biti)
D011	53265		VIC kontrol kaydı
		7	Raster karşılaştırması: (bit 8) 53266'ya bakınız
		6	Geliştirilmiş renk metni Mod: 1=aktif
		5	Bit harita modu: 1=aktif
		4	Sınır rengini sıfırla: 0=boşluk
		3	24/25 satır metin görüntüsü : 1=25 satır
		2-0	Y nokta-konumuna (0-7) düz kaydırma
D012	53266		Karşılaştırma IRQ'su için raster değeri okuma/ yazma
D013	53267		Işıklı-kalem mandal X konumu
D014	53268		Işıklı-kalem mandal Y konumu
D015	53269		Yaratık görünebilir: 1=aktif
D016	53270		VIC kontrol kaydı
		7-6	Kullanılmıyor
		5	Bu biti daima 0 yapın!
		4	Çok-renkli mod: 1=aktif (metin veya bit haritası)
		3	38/40 kolon metin görüntüsü: 1=40 kolon
		2-0	X konumuna duz kaydırma
D017	53271		0-7 yaratıkları 2X dikey büyütme (Y)
D018	53272		VIC bellek kontrol kayıtları
		7-4	Video matrisi taban adresi (VIC'in içinde)
		3-1	Karakter nokta-veri taban adresi (VIC'in içinde)
D019	53273		VIC kesinti bayrağı kaydı (bit=1: IRQ oluşumu)
		7	Herhangi bir açık IRQ koşulunda 1 olacak
		3	Işıklı-kalem tetikli IRQ bayrağı
		2	Yaratıklar arası çarpışma IRQ bayrağı
		1	Yaratık-zemin çarpışması IRQ bayrağı
		0	Raster karşılaştırması IRQ bayrağı

ADRES		BİT	TANIMLAMA
ONALTILIK	ONLUK		
D01A	53274		IRQ önleme kaydı 1=IRQ çalışabilir
D01B	53275		Yaratık-zemin görüntü önceliği: 1=yaratık
D01C	53276		Yaratık çok-renkli mod seçim: 1=çok-renkli mod
D01D	53277		0-7 yaratıkları 2X yatay büyütme (X)
D01E	53278		Yaratıklar arası çarpışmanın algılanması
D01F	53279		Yaratık zemin çarpışmasının algılanması
D020	53280		Çerçeve rengi
D021	53281		Zemin rengi 0
D022	53282		Zemin rengi 1
D023	53283		Zemin rengi 2
D024	53284		Zemin rengi 3
D025	53285		Yaratık çok-renkli kaydı 0
D026	53286		Yaratık çok-renkli kaydı 1
D027	53287		0'ıncı yaratığın rengi
D028	53288		1'inci yaratığın rengi
D029	53289		2'nci yaratığın rengi
D02A	53290		3'üncü yaratığın rengi
D02B	53291		4'üncü yaratığın rengi
D02C	53292		5'inci yaratığın rengi
D02D	53293		6'ncı yaratığın rengi
D02E	53294		7'nci yaratığın rengi
D400-D7FF	54272-55295		MOS 6581 ses ara birimi cihaz! (SID)
D400	54272		Ses 1: Frekans kontrolü. Küçük basamak baytı
D401	54273		Ses 1: Frekans kontrolü. Büyük basamak baytı
D402	54274		Ses 1: Darbe dalga genişliği. Küçük basamak baytı
D403	54275	7-4	Kullanılmıyor
		3-0	Ses 1=darbe dalga genişliği üst yarım bayt (High Nybble)
D404	54276		Ses 1: Kontrol kaydı
		7	Rastgele gürültü dalga biçimi. 1=Açık
		6	Darbe dalga biçimi seçimi. 1= Açık
		5	Dişli dalga biçimi seçimi. 1= Açık
		4	Üçgen dalga biçimi seçimi. 1= Açık
		3	Test biti: 1=Osilatör 1'in çalışmasını durdurur
		2	Osilatör 1 ve 3'ün modüle edilmesi. 1 = Açık
		1	Osc. 1'le Osc. 3 frekanslarının eş zamanlı hale getirilmesi. 1= Açık
		0	Geçit biti: 1=yükselme/düşme/durma durumu dizisine başlama. 0=kaybolma durumuna geçiş
D405	54277		Envelope üreteci 1: yükselme/düşme döngü kontrolü
		7-4	Yükselme döngüsü süresi seçimi: 0-15
		3-0	Yükselme döngüsü süresi seçimi: 0-15

ADRES		BİT	TANIMLAMA
ONALTILIK	ONLUK		
D406	54278		Envelope üreteci 1: Durma/kaybolma döngü kontrolü
		7-4	Durma döngü süresi seçimi: 0-15
		3-0	Kaybolma döngü süresi seçimi: 0-15
D407	54279		Ses 2: Frekans kontrol-küçük basamak baytı
D408	54280		Ses 2: Frekans kontrol-büyük basamak baytı
D409	54281		Ses 2: Darbe dalga biçimi genişliği. Üst yarım-bayt (High-Nybble)
D40A	54282	7-4	Kullanılmıyor
		3-0	Ses 2: Darbe dalga biçimi genişliği. Üst yarım-bayt
D40B	54283		Ses 2: Kontrol kaydı
		7	Rastgele gürültü dalga biçimi seçimi. 1=Açık
		6	Darbe dalga biçimi seçimi. 1=Açık
		5	Dişli dalga biçimi seçimi. 1=Açık
		4	Üçgen dalga biçimi seçimi. 1=Açık
		3	Test biti: 1=Osilatör 2 çalışma durumundan çıkar
		2	Osilatör 2 ve 1'in modüle edilmesi. 1=Açık
		1	Osilatör 2'nin osilatör 1 frekansı ile eş zamanlı hale getirilmesi. 1=Açık
		0	Geçit biti: 1=yükselme/düşme/durma durum dizisi başlangıcı. 0=kaybolma durumuna geçiş
D40C	54284		Envelope üreteci 2: Yükselme/düşme döngü kontrolü
		7-4	Yükselme döngü süresi seçimi: 0-15
		3-0	Düşme döngü süresi seçimi: 0-15
D40D	54285		Envelope üreteci 2: Durma/yok olma döngü kontrolü
		7-4	Durma döngü süresi seçimi: 0-15
		3-0	Kaybolma döngü süresi seçimi: 0-15
D40E	54286		Ses 3: Frekans kontrol-küçük basamak baytı
D40F	54287		Ses 3: Frekans kontrol-büyük basamak baytı
D410	54288		Ses 3: Darbe dalga biçimi genişliği-küçük basamak baytı
D411	54289	7-4	Kullanılmıyor
		3-0	Ses 3: Darbe dalga biçimi genişliği. Üst yarım-bayt (High-Nybble)
D412	54290		Ses 3: Kontrol kaydı
		7	Rastgele gürültü dalga biçimi seçimi. 1=Açık
		6	Darbe dalga biçimi seçimi. 1=Açık
		5	Dişli dalga biçimi seçimi, 1=Açık
		4	Üçgen dalga biçimi seçimi, 1=Açık
		3	Test biti: 1=Osilatör 3'ün durdurulması

ADRES			
ONALTILIK	ONLUK	BİT	TANIMLAMA
		2	Osilatör 3 ile 2'nin modüle edilmesi. 1=Açık
		1	Osilatör 3'ün osilatör 2 frekansıyla eş zamanlı hale getirilmesi. 1=Açık
		0	Geçit biti: 1=yükselme/düşme/durma durum dizisi başlangıcı. 0=kaybolma durumuna geçiş
D413	54291		Envelope üreteci 3: Yükselme/düşme/ durma döngü kontrolü
		7-4	Yükselme döngü süresi kontrolü: 0-15
		3-0	Düşme döngü süresi kontrolü: 0-15
D414	54292		Envelope üreteci 3: Durma/kaybolma döngü kontrolü
		7-4	Durma döngü süresi seçimi: 0-15
		3-0	Kaybolma döngü süresi seçimi: 0-15
D415	54293		Filtrenin geçiş frekansı: Alt yarım-bayt (Low-Nybble) (2-0 bitler)
D416	54294		Filtrenin geçiş frekansı: Büyük basamak baytı
D417	54295		Filtrenin salınım (rezonans) kontrolü/ses giriş kontrolü
		7-4	Filtre salınımı seçimi: 0-15
		3	Filtreye dışarıdan giriş: 1=Evet, 0=Hayır
		2	Ses 3 filtre çıkışı: 1=Evet, 0=Hayır
		1	Ses 2 filtre çıkışı: 1=Evet, 0=Hayır
		0	Ses 1 filtre çıkışı: 1=Evet, 0=Hayır
D418	54296		Filtre mod ve volum seçimi
		7	Ses 3 çıkışı kesimi: 1=Kapalı, 0=Açık
		6	Yüksek frekans geçişli filtre modu seçimi, 1=Açık
		5	Orta frekans geçişli filtre modu seçimi: 1=Açık
		4	Alçak frekans geçişli filtre modu seçimi: 1=Açık
		3-0	Çıkış ses volüm seçimi: 0-15
D419	54297		Örneksel/sayısal (A/D) çevirici: Game Paddle 1 (0-255)
D41A	54298		Örneksel/sayısal (A/D) çevirici: Game Paddle 2 (0-255)
D41B	54299		Osilatör 3 rastgele sayı üretici
D41C	54300		Ses üreteci 3 çıkışı
D500-D7FF	54528-55295		SID görüntüleri
D800-DBFF	55296-56319		Renk RAM'ı (Nybbles)
DC00-DCFF	56320-56575		MOS 6526 kompleks arabirim bağdaştırıcısı (CIA)#1
DC00	56320		Veri Portu A (Klavye, Joystick, Paddles, Light-Pen)
		7-0	Klavye taraması için, klavye kolon değerleri yazımı

ADRES		BİT	TANIMLAMA
ONALTILIK	ONLUK		
		7-6	Port A/B'de bulunan paddle'ların okunması (01=Port A, 10=Port B)
		4	Joystick A ateş düğmesi: 1=Ateş
		3-2	Paddle ateş düğmeleri
		3-0	Joystick A yönü (0-15)
DC01	56321		Veri Portu B (Klavye, Joystick, Paddles) Oyun Portu 1
		7-0	Klavye taraması için, klavye sıra değerleri okunması
		7	B zamanlayıcısı: Geçiş/vuruş çıkışı
		6	A zamanlayıcısı: Geçiş/vuruş çıkışı
		4	Joystick 1 ateş tuşu: 1=Ateş etmek
		3-2	Paddle ateş tuşları
		3-0	Joystick 1'in yönü
DC02	56322		Veri yön kaydı-Port A (56320)
DC03	56323		Veri yön kaydı-Port B (56321)
DC04	56324		Zamanlayıcı A: Alt bayt
DC05	56325		Zamanlayıcı A: Üst bayt
DC06	56326		Zamanlayıcı B: Alt bayt
DC07	56327		Zamanlayıcı B: Üst bayt
DC08	56328		Gün zamanı saati: 1/10 saniye
DC09	56329		Gün zamanı saati: Saniye
DC0A	56330		Gün zamanı saati: Dakika
DC0B	56331		Gün zamanı saati: Saat+öğleden önce/öğleden sonra (AM/PM) bayrağı (Bit 7)
DC0C	56332		Eş zamanı, veri g/ç veri tamponu
DC0D	56333		CIA kesinti kontrol kaydı (IRQ okuması/önleme yazımı)
		7	IRQ bayrağı (1=IRQ oldu)/IRQ yap/yapma bayrağı
		4	Bayrak 1 IRQ (kaset okuması/seri bağlantı SRQ girişi)
		3	Seri bağlantı yarıda kesme (interrupt)
		2	Gün zamanı saati alarm yarıda kesme
		1	B zamanlayıcısı yarıda kesme
		0	A zamanlayıcısı yarıda kesme
DC0E	56334		CIA A kontrol kaydı
		7	Gün zamanı saati frekansı: 1=50Hz, 0=60Hz
		6	Seri port g/ç modu. 1=Çıkış, 0=Giriş
		5	A zamanlayıcısı sayımı: 1=CNT sinyalleri, 0=Sistem 02 saati
		4	A zamanlayıcısı da yüklenecek mi? 1=Evet
		3	A zamanlayıcısı çatışma modu, 1=1-Vuruşluk, 0=Sürekli

ADRES		BİT	TANIMLAMA
ONALTILIK	ONLUK		
		2	A zamanlayıcısının PB6'ya çıkış modu: 1=Geçiş, 0=Vuruş
		1	A zaman/ayıcısının PB6'daki çıkışı: 1=Evet, 0=Hayır
		0	A zamanlayıcısının başlaması/durması 1=Başlama, 0=Durma
DC0F	56335		CIA kontrol kaydı B
		7	Alarm/TOD saati kurmak 1=Alarm, 0=Saat
		6-5	B zamanlayıcısının mod seçimi 00= Sistemdeki 02 saatinin vuruşlarını say 01=Pozitif CNT geçişlerini say 10= A zamanlayıcısı vuruş taşmalarını say 11=CNT pozitif iken A zamanlayıcısının taşmalarını say
		4-0	B zamanlayıcısı için CIA kontrol kaydı A ile aynısı
DD00-DDFF	56576-56831		MOS 6526 kompleks arabirim bağdaştırıcı (CIA) #2
DD00	56576		Veri Portu A (seri bağlantı RS-232, VIC bellek kontrolü)
		7	Seri bağlantı veri girişi
		6	Seri bağlantı saat vuruşu girişi
		5	Seri bağlantı veri çıkışı
		4	Seri bağlantı saat vuruşu çıkışı
		3	Seri bağlantı ATN çıkış sinyali
		2	RS-232 veri çıkış, (kullanıcı portu)
		1-0	VIC çipi sistem bellek bankası seçimi (varsayılan=11)
DD01	56577		Veri Portu B (kullanıcı portu, RS-232)
		7	Kullanıcı/RS-232 teyp hazır
		6	Kullanıcı/RS-232 gönderime hazır
		5	Kullanıcı
		4	Kullanıcı/RS-232 taşıyıcı tarama
		3	Kullanıcı/RS-232 halka göstergeç
		2	Kullanıcı/RS-232 veri terminali hazır
		1	Kullanıcı/RS-232 gönderim isteği
		0	Kullanıcı/RS-232 alınmış veri
DD02	56578		Veri yön kaydı-Port A
DD03	56579		Veri yön kaydı-Port B
DD04	56580		A zamanlayıcı alt bayt
DD05	56581		A zamanlayıcı üst bayt
DD06	56582		B zamanlayıcısı alt bayt
DD07	56583		B zamanlayıcısı üst bayt
DD08	56584		Gün saati: 1/10 saniye

ADRES			
ONALTILIK	ONLUK	BİT	TANIMLAMA
DD09	56585		Gün saati: Saniye
DD0A	56586		Gün saati: Dakika
DD0B	56587		Gün saati: Saat+AM/ PM bayrağı (bit 7)
DD0C	56588		Eş zamanlı Seri g/ç veri tamponu
DD0D	56589		CIA yarıda kesme kontrol kaydı (NMI okuma/ yazma maskesi)
		7	NMI bayrağı (1=NMI oluşumu)/bayrağı ayarla-temizle
		4	Bayrak 1 NMI (kullanıcı/RS-232 alınmış veri girişi)
		3	Seri port yarıda kesme
		1	B zamanlayıcısı yarıda kesme
		0	B zamanlayıcısı yarıda kesme
DD0E	56590		CIA kontrol kaydı A
		7	Gün saati frekansı: 1=50 Hz, 0=60 Hz
		6	Seri port g/ç modu: 1=Çıkış, 0=Giriş
		5	A zamanlayıcısı sayacı: 1=CNT sinyali, 0=Sistem 02 Saati
		4	A zamanlayıcısına zorunlu yükleme 1=Evet
		3	A zamanlayıcısına çalışma modu: 1=Tek atış, 0=Devamlı
		2	PB6'da A zamanlayıcısı çıkış modu: 1=Geçiş, 0=Vuruş
		1	PB6'da A zamanlayıcısı çıkışı: 1=Evet, 0=Hayır
		0	A zamanlayıcısı başlatma/durdurma, 1=Başlatma, 0=Durdurma
DD0F	56591		CIA kontrol kaydı B
		7	Alarm/TOD-saatini tanımla: 1=Alarm, 0=Saat
		6-5	B zamanlayıcısı mod seçimi: 00=Sistemdeki 02 saatinin vuruşlarını say 01=Pozitif CNT geçişlerini say 10=A zamanlayıcısı vuruş taşmalarını say 11=CNT pozitif iken A zamanlayıcısının taşmalarını say
		4-0	B zamanlayıcısı için CIA kontrol kaydı A ile aynısı
DE00-DEFF	56832-57087		Gelecekteki g/ç genişlemeleri için ayrılmıştır
DF00-DFFF	57088-57343		Gelecekteki g/ç genişlemeleri için ayrılmıştır

BÖLÜM 6

GİRİŞ/ÇIKIŞ REHBERİ

- Önsöz
- TV İçin Çıkış
- Diğer Cihazlar İçin Çıkış
- Oyun Portları
- RS-232 Arabirimi Açıklaması
- Kullanıcı Portu
- Seri Bağlantı
- Genişleme Portu
- Z-80 Mikroişlemci Kartuşu

ÖNSÖZ

Bilgisayarlar şu üç temel özelliğe sahiptirler: Hesap yapabilme, karar verebilme ve iletişim sağlayabilme. Programlanması en kolay olanı hesaplama işlemleridir. Matematik kurallarından çoğu bize yabancı değildir. Mantık kuralları çok az olduğundan, şimdilik bu kurallar hakkında fazla bilginiz olmasa bile, karar verme işlemini anlamanız çok zor değildir.

İletişimin, kural sayısı çok fazla olduğundan, yukarıda bahsedilen özellikler içinde en zorunun bu olduğunu söyleyebiliriz. Fakat bu durum, bilgisayarın tasarlanmasında yapılan bir dikkatsizlikten kaynaklanmaz. Kurallar, hemen hemen her cihazla iletişimi ve bu iletişimin birden fazla yolla yapılabilmesini sağlar. Esas kural şudur: Bilgi gönderenin, bilgiyi, alıcının anlayabileceği şekilde sunması gerekir.

TV İÇİN ÇIKIŞ

BASIC'te çıkışın en basit biçimi, PRINT yönergesidir. PRINT, TV ekranını çıkış cihazı olarak kullanır, bu durumda gözleriniz de ekrandaki bilgiyi kullandığı için giriş cihazı olarak düşünülebilirler.

Ekrana bilgi çıkışı yaparken esas amacınız, yazılan bilgiyi, kolay okunabilecek şekilde vermek olmalıdır. Bilgiyi en iyi şekilde yazabilmek için: renkleri kullanırken, harfleri yerleştirirken, küçük veya büyük harfleri yazarken veya grafik çizerken, kendinizi bir grafik sanatçısı gibi düşünmeye zorlamalısınız. Programınız ne kadar güzel yazılmış olursa olsun, sonuçların size ne vermek istediğini anlamanız gerektiğini unutmayın.

PRINT yönergesi bazı karakter kodlarını, takipçi için “komut” gibi kullanır. **CRSR** tuşu, gerçekte takipçinin yerini değiştirmekten başka hiçbir şey görüntülemez. Diğer komutlar, renk değişimi, ekranın temizlenmesi, boşluk yapımı ve araya karakter sokulması gibi işlemleri yerine getirir. **RETURN** tuşunun karakter kod numarası (CHR$) 13’tür. Bu kodların tam listesi EK C’de verilmiştir.

BASIC dilinde, PRINT yönergesi ile çalışan iki tane işlem vardır. TAB, takipçinin ekranın sol tarafından belli bir uzaklığa gelmesini sağlar. SPC ise takipçinin, bulunduğu yerden sağ tarafa doğru, verilen sayıda boşluk bırakarak, istenilen konuma gelmesi için kullanılır.

PRINT yönergesindeki noktalama işaretleri, bilgileri birbirinden ayırmak ve formatlamak için kullanılır. Noktalı virgül (;) işareti, 2 öğeyi aralarında hiç boşluk bırakmadan birbirinden ayırır. Satırın en sonunda ise, takipçi aşağıdaki satırın başına geçeceği yerde, yazılan son öğenin yanında kalır, yani RETURN tuşuna basılmamış gibi davranır.

Virgül (,) öğelerin sütunlar halinde ayrılmasını sağlar. Commodore 64'ün ekranında 10 karakterlik 4 kolon yer alır. Bilgisayarınız, bir virgül işareti gördüğünde, takipçiyi bir sonraki kolonun başına getirir. Satırın son kolonu da bittiğinde, takipçi bir sonraki satırın başına geçer. Noktalı virgülde de olduğu gibi satırın son öğesi ise, **RETURN** karakteri işleme sokulmaz.

Tırnak işaretleri (" "), yazılacak metinleri, değişkenlerden ayırmak için kullanılır. Satırdaki ilk tırnak işareti, metin alanının başlangıcını, son tırnak işareti de bitimini gösterir. Fakat şunu da belirtmeliyiz ki, satır sonuna tırnak işareti koymak zorunda değilsiniz.

RETURN kodu (CHR$ kodu 13), takipçinin bir sonraki satırın başına geçmesini sağlar. Bu, her zaman bir sonraki satır olmayabilir. Bir satır alta geçtikten sonra **RETURN**'e basıldığında, önceki satırla bağlantı gerçekleştirilmiş olur. Bilgisayar, bu iki satırın, gerçekte tek bir uzun satır olduğunu bilir. Bağlantılar, satır bağlantı tablosunda gösterilir (nasıl bağlantı kurulduğunu anlamak için bellek haritasına bakınız).

Mantıksal bir satır, neyin yazıldığına ve bastırıldığına bağlı olarak 1 ya da 2 ekran satırı uzunluğundadır. Takipçinin üzerinde bulunduğu mantıksal satır, **RETURN** tuşunun onu nereye göndereceğini belirler. Ekranın başlangıcında bulunan bir satır ise, ekranın, bir kerede 1 mi, yoksa 2 satır mı kayacağını gösterir.

TV ekranını çıkış cihazı olarak kullanmanın başka yolları da vardır. Grafikler konusunu işleyen bölüm, ekranda hareket edebilen cisimlerin yapılması ile ilgilidir. VIC çipi kısmı ise, ekran, sınır, renk ve boyutlarının değişimi konularını açıklar. Ses bölümünde de, TV hoparlörünün müzik ve ses efektleri yapmakta nasıl kullanıldığı anlatılır.

DİĞER CİHAZLAR İÇİN ÇIKIŞ

Genellikle, ekrandan ayrı olarak, teyp, yazıcı, disk sürücüsü veya modem gibi cihazlara bilgi çıkışı yapma ihtiyacı duyulur. BASIC'te bulunan OPEN yönergesi, bu bilgi çıkışının yapılabilmesi ve bir cihazın diğeri ile haberleşmesinin sağlanabilmesi için, kanal açmak amacıyla kullanılır. Kanal bir kez açıldığında, PRINT yönergesi ile, cihaza karakterler göndermek mümkündür.

OPEN ve PRINT# yönergeleri için örnek:

```
100 OPEN 4,4: PRINT# 4,"YAZICI ICIN CIKI
S"
110 OPEN 3,8,3,"0:DISK-DOSYA,S,W": PRINT
# 3,"DISKE GONDER"
120 OPEN 1,1,1,"TEYP-DOSYA": PRINT# 1,"T
EYBE YAZ"
130 OPEN 2,2,0,CHR$(10): PRINT# 2,"MODEM
E GONDER"
```

OPEN yönergesi, cihaz numarasına veya formatına göre her cihaz için değişiktir. Aşağıda değişik cihazlar için OPEN yönergelerinde kullanılan değişkenlerin listesi tablo halinde verilmiştir.

OPEN Yönergesinin Değişken Tablosu:

FORMAT: OPEN dosya-no, cihaz-no, sayı, yazınsal dizi

CİHAZ	CİHAZ NO	SAYI	YAZINSAL DİZİ
Kaset	1	0 = Giriş 1 = Çıkış 2 = EOT ile çıkış	Dosya ismi
Modem	2	0	Kontrol kayıtları
Ekran	3	0,1	
Yazıcı	4 veya 5	0 = Büyük harf veya grafikler 7 = Büyük veya küçük harfler	Yazılacak olan metin
Disk	8'den 11'e	2-14 = Veri kanalı 15 = Komut kanalı	Sürücü no, Dosya ismi, Dosya tipi, Okuma/yazma komutu

EOT: Kaset sonu (End Of Tape)

YAZICI İÇİN ÇIKIŞ

Yazıcı, ekran ile aynı tipte bir çıkış cihazıdır. Bilgi gönderilirken yapmanız gereken en önemli iş, gözle kolay algılanabilen bir biçim bulmanızdır. Bu iş için kullanabilecekleriniz: ters, çift-genişlikli, büyük ve küçük harfler ile programlanabilen-nokta grafikleridir. SPC işlevi ekrandakinin aynısıdır. Buna karşılık, TAB fonksiyonu takipçinin, kâğıtta değil de ekran satırındaki konumunu hesapladığı için, doğru olarak çalışmaz.

YAZICI için OPEN yönergesi, iletişim kanalı açmakta kullanılır. Ayrıca hangi karakter setinin kullanılacağını belirler (Grafik, Büyük veya Küçük harfler).

YAZICI için OPEN yönergesi örnekleri:

```
OPEN 1,4: REM BUYUK HARF/GRAFIK
```

```
OPEN 1,4,7: REM BUYUK VE KUCUK HARF
```

Bir karakter seti ile çalışırken, bazı satırları karşıt karakter setini kullanarak bastırmak mümkündür. Grafik ve Büyük harf setini kullanırken, takipçi-aşağıya karakteri (CHR$(17)), büyük ve küçük harf setine geçmenizi aynı şekilde; büyük ve küçük harf setini kullanırken de takipçi-yukarı (CHR$(145)) karakteri, büyük harf ve grafik setine geçmenizi ve karakterleri bu seti kullanarak bastırmanızı sağlar.

Yazıcı için diğer özel fonksiyonlar, karakter kodları ile kontrol edilir. Bu kodları, diğer karakterler gibi basit bir şekilde bastırmak mümkündür.

Yazıcı Kontrol Karakter Kodları Tablosu:

CHR$ KOD	AMACI
10	"Line feed" ("satır başı")
13	RETURN (CBM yazıcılarında otomatik satır başı)
14	Çift-genişlikli karakter moduna giriş
15	Çift-genişlikli karakter modundan çıkış
18	Negatif karakter moduna giriş
146	Negatif karakter modundan çıkış
17	Büyük/küçük harf karakter setine giriş
145	Büyük harf/grafik karakter setine giriş
16	Arkadan gönderilen 26 baytlık karakterlerle gösterilen pozisyona TAB
27	Belirlenen noktaya ilerlemesi
8	Programlanabilir-Nokta grafik moduna giriş
26	Grafiklerin tekrarı

Komut kodlarının kullanımı için COMMODORE YAZICISININ ELKİTABI'na bakınız.

MODEM İÇİN ÇIKIŞ

Modem, karakter kodlarını duyulabilir vurumlara çeviren veya bu işlemin tersini yapan basit bir cihazdır. Böylece bilgisayarlar, telefon hatlarını kullanarak haberleşebilirler. Modem için kullanılan bir OPEN yönergesi, iletişime girilecek diğer bilgisayarla hız ve biçim uyumunun sağlanması için, değişkenlerin ayarlanmasında kullanılır. OPEN yönergesinin sonunda, iki karakterlik, bir yazınsal dizi gönderilebilir.

İlk karakter kodundaki bit konumları, baud hızını, veri bitlerinin ve stop bitlerinin sayısını belirler. İkinci kodu kullanıp kullanmamak size bırakılmıştır. Bu kod, gönderimin çift yönlülüğünü ve eşliğini belirlemek için kullanılır. Bu cihaz hakkındaki daha ayrıntılı bilgileri VICMODEM elkitabının RS-232 bölümünde bulabilirsiniz.

MODEM için OPEN yönergesi örneği:

```
OPEN 1,2,0,CHR$(6): REM 300 BAUD
```

```
100 OPEN 2,2,0,CHR$(163) CHR$(112): REM
BAUD, V.S.
```

Birçok bilgisayar bilgi alışverişi için ASCII diye bilinen Amerikan Standart Kodunu kullanır (ASKİİ olarak telaffuz edilir). Bu standart karakter kod seti, Commodore 64'te kullanılandan farklıdır. Diğer bilgisayarlarla yapılan iletişimlerde, Commodore karakter kodlarının ASCII karşılıklarına çevrilmesi gerekir. Standart ASCII kodlarının bir tablosu, kitapta Ek C kısmında verilmiştir.

Karakterlerin çevrilmesi dışında, modeme çıkış fazla karışık bir iş değildir. Buna karşılık bilgisayarınızın insan eli değmeden, bir başka bilgisayarla haberleşmesini sağlayan programlar yazmak için, karşıdaki cihazın çalışma sistemini çok iyi bilmek gerekir. Buna örnek olarak, hesap numaranızı ve şifrenizi otomatik olarak yazan terminal programını verebiliriz. Bu işin en iyi şekilde yapılabilmesi için karakterin ve RETURN karakterlerinin sayısını dikkatlice hesaplamanız gerekir. Aksi takdirde bilgisayar, karakterleri algıladığında, onlarla ne yapacağını bilemez.

TEYP ÜNİTESİ İLE ÇALIŞMA

Kaset bantlarının, veri için sınırsız kapasiteleri vardır. Uzun bir bant, içinde daha fazla bilgi saklayabilecektir. Fakat bantlar, zaman açısından sınırlandırılmışlardır. Bandın uzun olması, aranan bilginin daha uzun zamanda bulunmasına neden olacaktır.

Bant kullanan bir programcının, zaman faktörünü en aza indirgemeye çalışması gerekir. Bu işlem için genelde kullanılan metot, tüm kaset veri dosyasını RAM'e okumak, işlemek ve ondan sonra tüm veriyi tekrar kasete saklamaktır. Bu işlem size, verilerinizi sıralama, düzenleme ve inceleme olanağı verecektir. Fakat bu da dosyanızı, RAM'in kapasitesinin elverdiği ölçüde sınırlamanızı gerektirecektir.

Dosyanızın büyüklüğü, RAM'de elverişli olan alandan büyükse, artık disket kullanmanın zamanı gelmiştir. Disket, içinde saklı bulunan verilerin hepsinin okunmasına gerek duyulmaksızın, yalnızca istenilen kısmın okunabilmesini sağlar. Ayrıca dosyanızın tümünü değiştirmeden, eski bir verinin üstüne yenisini yazabilirsiniz. Hesap defteri tutulması ve faturalama gibi iş uygulamaları için disket kullanılmasının nedeni budur.

PRINT# yönergesi, PRINT'in yaptığı gibi, veriyi biçimlendirir. Bütün noktalama işaretlerinin işlevi aynıdır. Fakat, artık ekranla çalışmadığınızı unutmayın. Formatlama, INPUT# yönergesiyle birlikte yapılmalıdır.

PRINT#1, A$, B$, C$ yönergesini düşünelim. Ekranda kullanıldığında, değişkenler arasındaki virgül, onları aralarında gerekli boşlukları bırakarak, 10 karakter genişliğinde kolonlar halinde ekrana yerleştirecektir. Kasette, yazınsal dizinin uzunluğuna göre 1'den 10'a kadar bir boşluk eklenecektir. Bu da kasetinizin boşa harcanmasına neden olur.

Daha kötüsü ise, INPUT# yönergesinin, bu yazınsal diziyi okumaya çalışmasında olanlar. INPUT#1, A$, B$, C$ yönergesi B$ ve C$ için hiçbir veri bulamayacaktır. Onlar yerine A$, boşluklarla birlikte tüm verileri içerecektir. Neler olduğunu herhalde anlayamadınız. Şimdi bir bant dosyasına bakalım.

```
A$="KOPEK" B$="KEDI" C$="AGAC"
PRINT#1, A$, B$, C$
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
K O P E K           K E D I                     A G A C RETURN
```

INPUT# yönergesi, bildiğiniz INPUT yönergesi gibi çalışır. INPUT yönergesi için veri yazılırken, veri öğeleri ya **RETURN** tuşuna basılarak ya da aralarına virgül konarak birbirlerinden ayrılır. PRINT# yönergesi yine aynı PRINT yönergesinde olduğu gibi, satır sonuna RETURN koyar. A$'ın üç değer birden almasının nedeni ise, en sonundakinin dışında, aralarında hiçbir ayırıcının bulunmamasıdır.

Bant için uygun bir ayırıcı virgül (,) veya RETURN olabilir. RETURN kodu, PRINT ya da PRINT# yönergelerinin sonuna otomatik olarak konur. RETURN kodunu, her iki öğenin arasına koyabilmek için bir yöntem de her öğe için bir PRINT # yönergesi kullanmaktır. Daha iyi bir yöntem ise bir değişkeni, RETURN CHR$ kodu olan CHR$(13) olarak tanımlamak veya virgül kullanmaktır. Bu işlem için yazılacak yönerge R$=",":PRINT#1, A$ R$ B$ R$ C$ olacaktır. Değişken isimleri arasında virgül veya herhangi bir noktalama işareti kullanmayınız. Çünkü Commodore 46 onları ayrı tutacak ve programınızda sadece yer kaplayacaklardır.

Uygun bir kaset dosyasının görünüşü aşağıdaki gibi olabilir:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
K O P E K , K E D I  ,  A  G  A  C  RETURN
```

GET# yönergesi, kasetten bir defada bir karakter okur. Her karakteri, hatta RETURN kodunu ve noktalama işaretini de okuyacaktır. CHR$(0) kodu, 0 kodlu bir karakter dizisi olarak değil, boş bir dizi olarak okunur. ASC fonksiyonunu, boş bir dizi için kullanmak isterseniz, ILLEGAL QUANTITY ERROR (geçersiz miktar hatası) uyarısı ile karşılaşırsınız.

GET#1, A$: A=ASC(A$) satırı programlarda, genellikle kasetteki verileri incelemek için kullanılır. Hatta uyarılarından kaçınmak için, satırın GET#1, A$: A= ASC(A$+CHR$(0)) olarak değiştirilmesi gerekir. Sondaki CHR$(0), boş diziler için sigorta olarak kullanılır. Fakat A$'da başka karakter olduğu zaman etkisi yoktur.

FLOPPY DİSKETLERDE VERİ SAKLAMA

Disketler, verilerin 3 değişik şekilde saklanmasına izin verirler. Sequential (sıralı) dosyalar, kasettekilere benzer. Fakat bunların birçoğu, aynı zamanda kullanılabilir. Relative (göreli) dosyalar, verinin kayıtlar halinde tutulmasına ve bunların daha sonra birer birer okunabilmesine ve değiştirilebilmesine olanak verir. Random (rasgele) dosyalar ise, disketin herhangi bir kısmındaki veriler ile çalışmanızı sağlar. Blok denilen, 256 baytlık bölümler halinde organize edilmişlerdir.

PRINT# yönergesinin kısıtlamaları daha önce, kasetlerle ilgili bölümde tartışılmıştı. Formatlama ile ilgili bazı kısıtlamalar, disket için de geçerlidir. Verileri ayırmak için RETURN veya virgül kullanılması gereklidir ve CHR$(0) da, yine GET# yönergesiyle boş dizi olarak okunur.

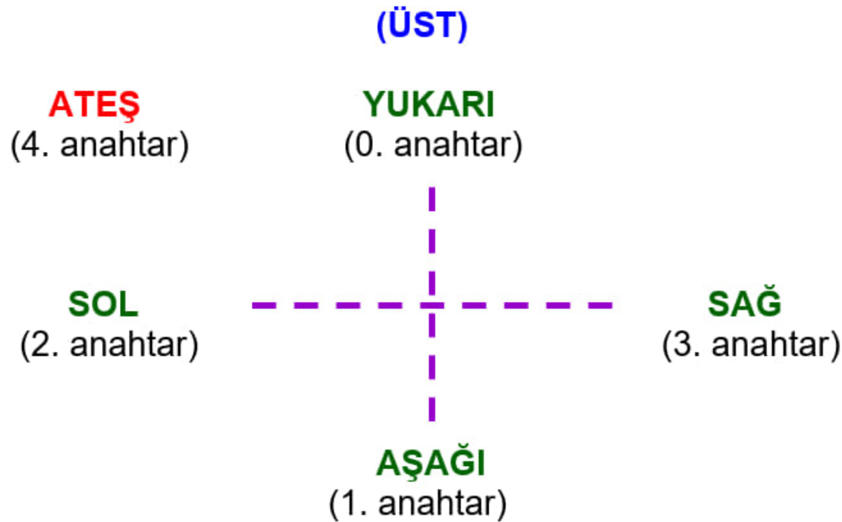
Relative ve Random dosyaların ikisi de ayrı veri ve komut kanallarını kullanır. Diskete kaydedilebilecek olan veri, disk RAM'inin içinde bulunan geçici depoya (buffer) yerleştirilmek üzere veri kanalına gider. Kayıt ya da blok tamamlandığında sürücüye komut kanalından, verinin nereye konulacağını belirten bir komut gönderilir ve tüm deponun içindekiler yazılır. Büyük miktarlarda verilerin işlenmesini gerektiren uygulamalar, Relative dosyalara saklanmalıdır. Böylece en kısa zamanda işlem görülecek ve programcıya çok büyük bir esneklik sağlamış olacaktır. Disk Sürücü Elkitabı, disk dosyaları kullanımı için tam bir programlama rehberidir.

OYUN PORTLARI

Commodore 64'te: oyun-kolu, kürek ve ışık-kalem kullanabilmenizi sağlayan iki tane 9 uçlu oyun portu vardır. Her port ya bir oyun-kolu ya da bir kürek çiftini kabul eder. Özel grafik kontrolü gibi işlemler için port A'ya ışık-kalem takılabilir. Bu bölüm size, oyun-kolu ve küreklerin BASIC ve makine dilindeki kullanımı hakkında örnekler verecektir.

Dijital oyun-kolu, CIA#1'e bağlanır (MOS 6526 Complex Interface Adapter). Bu giriş/çıkış cihazı, ayrıca kürek ateşleme düğmeleri ve klavye tarama işlerini de üstlenmiştir. 6526 CIA çipi, 56320'den 56335'e ($DC00'dan $DC0F'e) kadar olan bellek yerleşimlerinde yer alan 16 kayda sahiptir. Port A 56320 ($DC00)'de Port B ise 56321 ($DC01)' de yer alır.

Bir dijital oyun-kolunun 4'ü yön için, 1'i ateşleme için kullanılan 5 anahtarı vardır. Oyun-kolu anahtarları aşağıdaki gibi düzenlenmiştir;



Bu anahtarlar, 56320 veya 56321 'deki en alt 5 bite karşılık gelir. Normal olarak, bir yön SEÇİLMEDİĞİNE veya ateşleme düğmesine BASILMADIĞINDA, karşılık gelen bit bir olur. Ateşleme düğmesine basıldığında, bit (bu durumda 4'üncü bit) 0 olur. BASIC'ten oyun-kolunun okunması için aşağıdaki altprogramın bir benzerinin kullanılması gerekir:

```
10 FORK=0TO10:REM YON DIZISINI AYARLA
20 READDR$(K):NEXT
30 DATA"","K","G","","B","KB"
40 DATA"GB","","D","KD","GB"
50 PRINT"GIT.....";
60 GOSUB100:REM OYUN-KOLU OKU
65 IFDR$(JV)=""THEN80:REM YONUN SECILIP
SECILMADIGINI KONTROL ET
70 PRINTDR$(JV);" ";:REM CIKIS HANGI YON
80 IFFR=16THEN60:REM ATES DUGMESINE BASI
LI OLUP OLMADIGINI KONTROL EDIN
90 PRINT"-----A-----T-----E-----S-----
!!!":GOTO60
100 JV=PEEK(56320):REM OYUN-KOLU DEGERIN
I AL
110 FR=JVAND16:REM ATES DUGMESI DURUMU
120 JV=15-(JVAND15):REM YON DEGERI
130 RETURN
```

> **NOT:** İkinci oyun-kolu için JV=PEEK (56321) olmalıdır.

JV değerleri şu yönlere karşılık gelir:

JV EŞİTTİR	YÖN
0	Hiçbiri
1	Yukarı
2	Aşağı
3	-
4	Sola
5	Yukarı ve sola
6	Aşağı ve sola
7	-
8	Sağa
9	Yukarı ve sağa
10	Aşağı ve sağa

Aynı görevi yapan, kısa bir makine dili programı aşağıda verilmiştir.

```
1000 . SAYFA (OYUN-KOLU.8/5) OYUN-KOLU D
UGME OKUMA RUTINI
1010 ;
1020 ;
1030 ;
1040 DX = $C110
1050 DY = $C111
1060 * = $C200
1070 DJRR LDA $DC00 ; YALNIZ PORT A'DAN
VERI AL
1080 DJRRB LDY #0   ; BU RUTIN AKUMULATO
RDE YER ALAN
1090 LDX #0         ; OYUN-KOLUNUN ATESL
EME TUSU ILE
1100 LSR A          ; ILGILI VERILERI OK
UR VE COZER. BU EN SAGDAKI 5 BITTE
1110 BCS DJR0       ; ANAHTARIN DURUMU I
LE ILGILI BILGILER BULUNUR.
1120 DEY            ; EGER ANAHTAR KAPAL
I ISE BIR BIT SIFIR
1130 DJR0 LSR A     ; EGER ANAHTAR ACIK
1140 BCS DJR1       ; BIR BIT BIR OLUR
1150 INY            ; OYUN-KOLUNUN YONLE
RI SAG, SOL, ASAGI, YUKARIDIR.
1160 DJR1 LSR A     ; BIT3=SAG, BIT2=SOL
, BIT1=ASAGI, BIT0=YUKARI
1170 BCS DJR2       ; VE BIT4=ATES TUSU.
1180 DEX            ; RTS ZAMANINDA YON
NUMARALARI IKI'NIN
1190 DJR2 LSR A     ; KOMPLEMENTI SEKLIN
DE JX VE DY'DE
```

```
1200 BCS DJR3    ; YER ALIRLAR. ORNEK DX=
1 (SAGA GIT)
1210 INX         ; DX=-1 (SOLA GIT). Dx=0
(X DEGISMIYOR)
1220 DJR3 LSR A ; DY=-1 (EKRANDA YUKARI
GIT), DY=0 (Y DEGISMIYOR).
1230 STX DX      ; OYUN-KOLU YUKARI, EKRA
NIN YUKARI KISMINA GITMESINI
CORRESPONDS
1240 STY DY      ; OYUN-KOLU ASAGI ISE, E
KRANIN ASAGI KISMINA GITMESINI
1250 RTS         ; IFADE EDER.
1260             ;
1270             ; RTS ZAMANINDA, ATES TU
SUNUN DURUMU TASMA BAYRAGINDA YER ALIR.
1280             ; EGER C=1 ISE TUSA BASI
LMISTIR. EGER C=0 ISE TUSA BASILMAMISTIR
1290 ;
1300 .END
```

KÜREKLER (PADDLE)

Kürek, CIA#1 ve SID (MOS 6581 Sound Interface Device) çipi oyun portunu kullanarak bağlanır. Kürek değeri, SID yardımıyla 54297 ($D419) ve 54298 ($D41A) dan okunur. KÜREK DEĞERLERİ SADECE BASIC'TEN OKUNDUĞUNDA, GÜVENİLİR OLMAZ!.. Kürekleri BASIC veya makine kodundan kullanmanın en iyi yolu, aşağıdaki makine dili altprogramını kullanmaktır. (BASIC'den SYS komutunu kullanarak çalıştırın ve sonra altprogramın kullandığı bellek yerleşimlerini PEEK komutu ile kontrol edin).

```
1000 ;*********************************
1010 ;* DORT KUREK OKUMA RUTINI (IKI    *
1020 ;* ICINDE KULLANILABILR)           *
1030 ;*********************************
1040 PORTA=$DC00
1050 CIDDRA=$DC02
1060 SID=$D400
1070 *=$C100
1080 BUFFER *=*+1
1090 PDLX *=*+2
1100 PDLY *=*+2
1110 BTNA *=*+1
1120 BTNB *=*+1
1130 * = $C000
1140 PDLRD
1150 LDX #1      ; DORT KUREK VEYA IKI AN
ALOG OYUN-KOLU ICIN
1160 PDLRD0      ; BIR CIFT ICIN GIRIS NO
KTASI (BIRINCI X DURUMU)
1170 SEI
1180 LDA CIDDRA ; DDR'NIN O ANSAKI DEGER
INI AL
```

```
1190 STA BUFFER  ; SAKLA
1200 LDA #$C0
1210 STA CIDDRA  ; PORT A'YI GIRIS ICIN
TANIMLA
1220 LDA #$80
1230 PDLRD1
1240 STA PORTA   ; BIR CIFT KUREGE ADRES
VERIN
1250 LDY #$80    ; BIRAZ BEKLE
1260 PDLRD2
1270 NOP
1280 DEY
1290 BPL PDLRD2
1300 LDA SID+25  ; X DEGERINI AL
1310 STA PDLX,X
1320 LDA SID+26  ; Y DEGERINI AL
1330 STA PDLY,X
1340 LDA PORTA   ; KUREGIN ATES TUSUNU O
KUMAK ICIN SURE VER
1350 ORA #80     ; DIGER CIFT ICIN NE YA
PTIYSAN ONU YAP
1360 STA BTNA    ; BIT 2 PDL X, BIT 3 IS
E PDL Y'DIR
1370 LDA #$40
1380 DEX         ; TUM CIFTLER TAMAM MI?
1390 BPL PDLRD1  ; HAYIR
1400 LDA BUFFER
1410 STA CIDDRA  ; DDR'IN ONCEKI DEGERIN
I GERI YUKLE
1420 LDA PORTA+1 ; IKINCI CIFT ICIN
1430 STA BTNB    ; BIT 2 PDL X, BIT 3 IS
E PDL Y'DIR
1440 CLI
1450 RTS
1460 .END
```

Kurekler aşağıdaki BASIC programı yardımıyla okunabilir:

```
10 C=12*4096:REM KUREGIN RUTIN BASLANGIC
INI TANIMLA
15 FORI=0 TO 63:READ A:POKE C+I,A: NEXT
20 SYSC:REM KUREGIN RUTINI CAGIR
30 P1=PEEK(C+257):REM KUREGIN BIR DEGERI
NI AYARLA
40 P2=PEEK(C+258):REM KUREGIN IKI DEGERI
NI AYARLA
50 P3=PEEK(C+259):REM KUREGIN UC DEGERIN
I AYARLA
60 P4=PEEK(C+260):REM KUREGIN DORT DEGER
INI AYARLA
61 REM ATES TUSUNUN DURUMUNU OKU
62 S1=PEEK(C+261):S2=PEEK(C+262)
70 PRINTP1,P2,P3,P4:REM KUREGIN DEGERINI
BAS
```

```
72 REM ATES TUSLARININ DURUMUNU BAS
75 PRINT:PRINT"ATES A ";S1,"ATES B ";S2
80 FORW=1TO50:NEXT:REM BIRAZ BEKLE
90 PRINT"♥":PRINT:GOTO20:REM EKRANI TEMI
ZLE VE TEKRAR YAP
95 REM MAKINE KODU RUTUNI ICIN VERI
100 DATA 162,1,120,173,2,220,141,0,193,1
69,192,141,2,220,169
110 DATA 128,141,0,220,160,128,234,136,1
6,252,173,25,212,157
120 DATA 1,193,173,26,212,157,3,193,173,
0,220,9,128,141,5,193
130 DATA 169,64,202,16,222,173,0,193,141
,2,220,173,1,220,141
140 DATA 6,193,88,96
```

IŞIK-KALEMİ (LIGHT-PEN)

Işık-kalemi, sinyal aşağı geçişi sırasında o anki ekran konumunu bir çift kayda yerleştirir (LPX, LDY). X konum kaydı 19($13). X konumunun geçiş sırasındaki en soldaki 8 bitini içerir. X konumu bir 512-durum sayacı (9 bit) tarafından tanımlandığından 2 yatay nokta hassasiyeti sağlanmıştır. Buna benzer şekilde, Y konumu kendi 20($14) kaydına yazılır, fakat burada 8 bit, görünebilir kısımda, tek ızgara hassasiyeti sağlar. Işık-kalemi mandalı her çerçevede sadece bir kez tetiklenebilir ve bundan sonraki tetiklenmelerin etkisi yoktur. Öyleyse işlemden önce ışık-kaleminizin karakteristiklerine göre, ışık-kaleminizi ekrana çevirmeden önce birkaç örnekleme almalısınız (ortalama 3 ya da daha fazla).

RS-232 ARABİRİMİ AÇIKLAMASI

GENEL TASLAK

Commodore 64'ün içinde, RS-232 modem, yazıcı veya başka bir cihazla bağlantı kurmak için hazır bir RS-232 arabirimi vardır. Commodore 64'e bir cihaz bağlamak için tek gereksiniminiz, bir kablo ve az bir programlama bilgisidir.

RS-232 Commodore 64'te, standart RS-232 formatı ile kurulur, fakat kullanılan voltajlar, RS-232'nin normal olarak -12 ve 12 arasındaki voltajlar değil, TTL seviyeleridir (0-5V). Commodore 64'le RS-232 cihazı arasındaki kablonun voltaj değişikliğine uygun olması gerekir. Commodore'un RS-232 arabirim kartuşu bu sorunu halledecektir. Makine dili ile programlamak için RS-232 arabiriminin yazılmasına BASIC'ten veya KERNAL'dan erişmek mümkündür.

BASIC seviyesinde RS-232, normal BASIC komutları kullanır: OPEN, CLOSE, CMD, INPUT#, GET#, PRINT#, ST değişkeni, alıcı tamponundan veri alınıp getirilmesi için INPUT# ve GET#, gönderici tamponuna veri yerleştirilmesi için PRINT# ve CMD. Bu komutların kullanımı (ve örnekleri) hakkında detaylı bilgi, daha sonra bu bölümde verilecektir.

RS-232 KERNAL bayt ve bit seviye taşıyıcıları, 6526 CIA#2 cihazının zamanlayıcı ve kesintilerinin kontrolü altında çalışır. 6526 çipi, RS-232 işlemi için NMI (önlenemez-kesinti) rutinleri üretir. Bu istekler, BASIC ve makine dili programlama sırasında arka planda. RS-232 işlemlerinin yapılmasını sağlar. RS-232 rutinleri tarafından üretilen NMI'lar ile verilerin saklanması ve gönderilmesi sırasında kesinti olmasını önlemek için, KERNAL, kaset ve seri veri yolu rutinlerinde hazır bloke eden yapılar (hold-off) vardır. Kaset ve seri veri yolu aktiviteleri sırasında, RS-232 cihazlarından veri ALINAMAZ. Fakat bu bloke eden yapılar yalnızca belirli yerlere ait olduklarından (sizin programlama konusunda titiz davrandığınız varsayılarak) herhangi bir karışıklık meydana gelmez.

Commodore 64 RS-232 ara biriminde, RS-232 bilgilerinin gönderilmesi ve alınması sırasında veri kaybının önlenmesine yardımcı, iki depo vardır.

Commodore 64 RS-232 KERNAL depoları, belleğin başlangıcında, her biri 256 bayt uzunluğundaki ilk giren-ilk çıkar (FIFO) özelliği taşıyan iki depodan oluşur. RS-232 kanalı açıldığında (OPEN ile), belleğin 512 baytı bu depolar için otomatik olarak ayrılır. Eğer BASIC programınızdan sonra bellekte yeterli miktarda boş yer yoksa. Herhangi bir hata mesajı basılmaz ve programınızın son kısmı tahrip edilir. BU YÜZDEN, DİKKATLİ OLUNUZ.

CLOSE komutunu kullandığınızda bu tamponlar otomatik olarak iptal edilir.

RS-232 KANALININ AÇILMASI

İkinci bir OPEN, yönergesi depo göstergeçlerini sıfırlayacağından, her zaman sadece bir RS-232 kanalı açılmalıdır. Aksi takdirde gönderici veya alıcı depolardaki karakterler kaybolabilir.

Dosya ismi alanında, en fazla 4 karakter gönderilebilir. İlk ikisi kontrol ve komut kaydı karakterleri, diğer ikisi, gelecekteki sistem seçenekleri olarak ayrılmıştır. Baud hızı, eşlik ve diğer değişkenler burada tanımlanabilir.

Kontrol sözcüğünde hatalı baud hızının bulunmasına yönelik hiçbir hata kontrolü yapılmaz. Herhangi bir kuraldışı kontrol sözcüğü, sistem çıkışının çok yavaş bir hızla yapılmasına neden olur (50 baud'tan az).

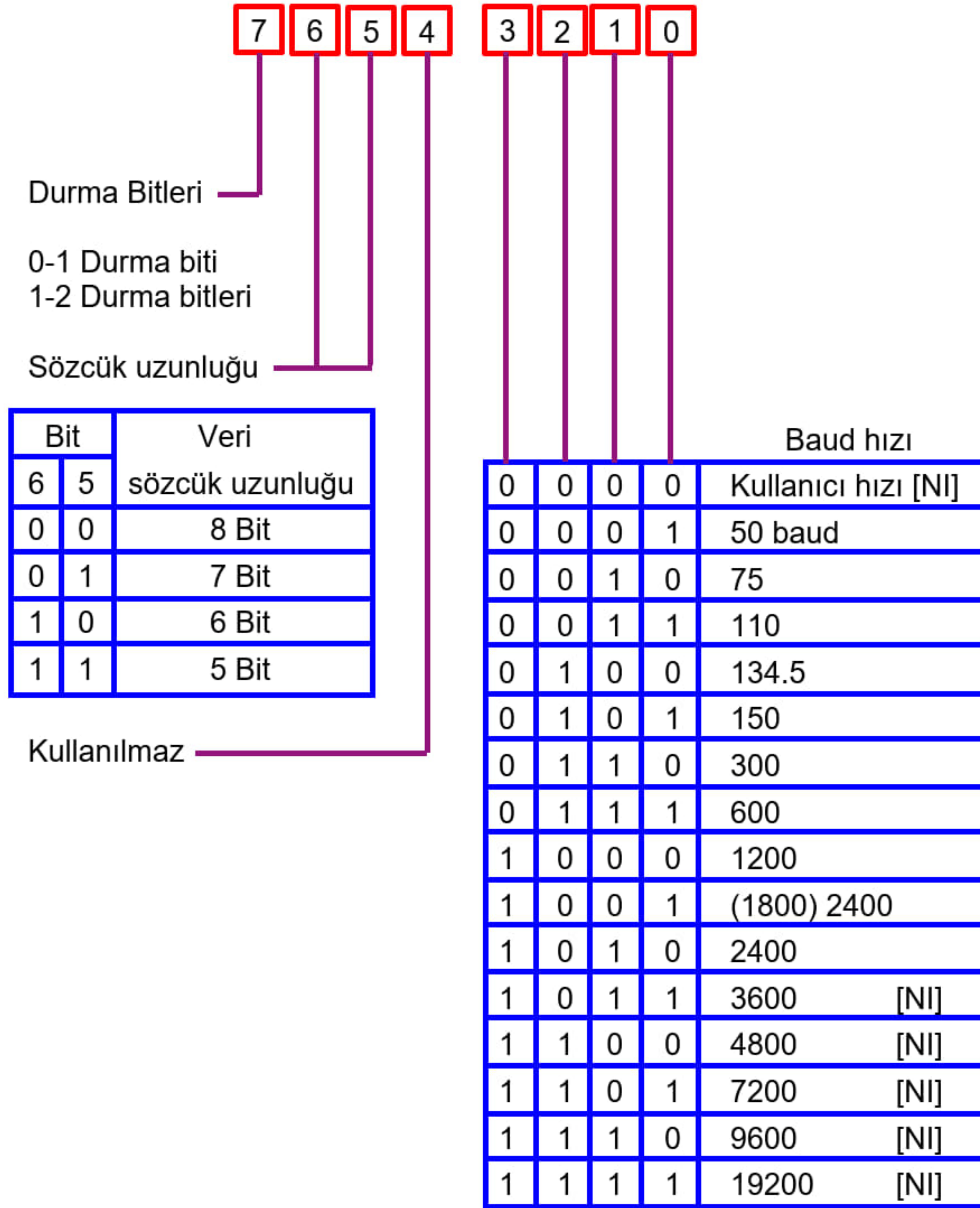
BASIC YAZIMI

OPEN mdn,2,0,"<kontrol kaydı> <komut kaydı> < keyfi baud hızı, alt> <keyfi baud hızı, üst >"

Mdn = Mantıksal dosya numarası

Mantıksal dosya numarası 1'den 255'e kadar herhangi bir sayı olabilir. Fakat 121'den büyük bir mantıksal dosya numarası seçtiğinizde bütün satır başlarını (carriage return) bir satır ilerlemesinin (line feed) takip edeceğini unutmayınız.

Kontrol kaydı haritası şekil 6-1'de aşağıda verilmiştir. Bu şekil üzerinde bir değerleri, veri sözcük uzunlukları, baud hızı ve kullanıcı hızı (NI) ayrıntılı bir biçimde gösterilmektedir.



Bit		Veri
6	5	sözcük uzunluğu
0	0	8 Bit
0	1	7 Bit
1	0	6 Bit
1	1	5 Bit

3	2	1	0	Baud hızı
0	0	0	0	Kullanıcı hızı [NI]
0	0	0	1	50 baud
0	0	1	0	75
0	0	1	1	110
0	1	0	0	134.5
0	1	0	1	150
0	1	1	0	300
0	1	1	1	600
1	0	0	0	1200
1	0	0	1	(1800) 2400
1	0	1	0	2400
1	0	1	1	3600 [NI]
1	1	0	0	4800 [NI]
1	1	0	1	7200 [NI]
1	1	1	0	9600 [NI]
1	1	1	1	19200 [NI]

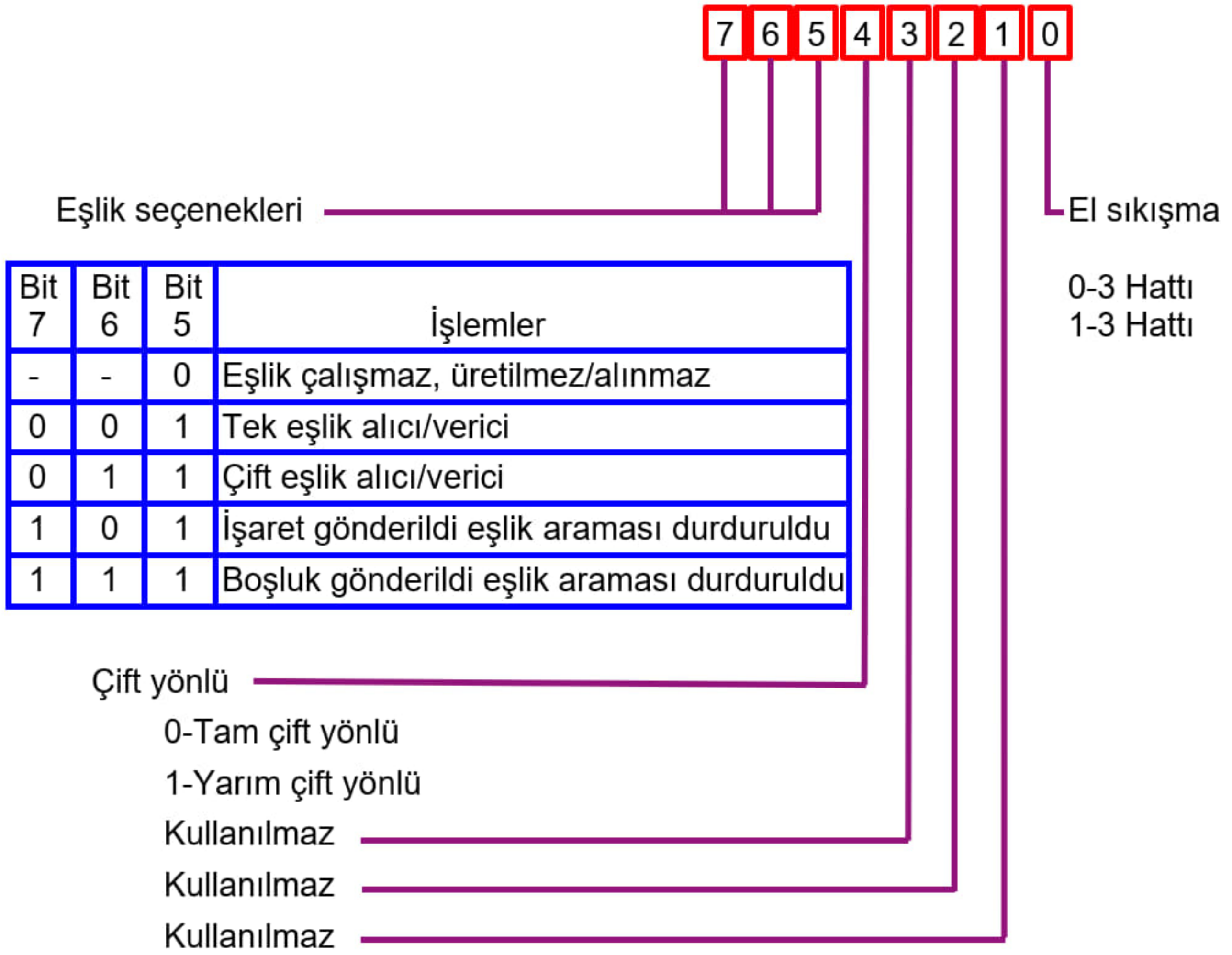
Şekil 6-1 Kontrol Kaydı Haritası

<kontrol kaydı> - baud hızını belirlemek için gerekli olan tek baytlık bir karakterdir (Şekil 6-1'e bakın).

Baud hızının en son 4 biti 0 ise, <keyfi baud hızı alt> <keyfi baud hızı üst> karakterleri size aşağıdakilere bağlı bir hız verecektir:

<keyfi baud hızı alt >=<sistem frekansı>/hız/2-100-<keyfi baud hızı üst>*256

<keyfi baud hızı üst>= INT((<sistem frekansı>/hız/2-100)/256)



Bit 7	Bit 6	Bit 5	İşlemler
-	-	0	Eşlik çalışmaz, üretilmez/alınmaz
0	0	1	Tek eşlik alıcı/verici
0	1	1	Çift eşlik alıcı/verici
1	0	1	İşaret gönderildi eşlik araması durduruldu
1	1	1	Boşluk gönderildi eşlik araması durduruldu

Şekil 6.2 Komut kaydı Haritası

Yukarıdaki formüller şu temele dayanır:

Sistem Frekansı = 1.02273E6 NTSC (Kuzey Amerika TV standarttı)
= 0.98525E6 PAL (U.K. ve çoğu Avrupa TV standarttı)

<komut kaydı> - Diğer terminal değişkenlerini açıklayan tek baytlık karakterdir (Şekil 6-2'ye bakın). Bu karakter GEREKLİ DEĞİLDİR.

KERNAL GİRİŞİ

OPEN ($FFC0) (Giriş komutları ve durumları hakkında daha fazla bilgi edinmek istiyorsanız KERNAL tarifnamesine bakınız).

ÖNEMLİ NOT: Bir BASIC programında, RS-232 OPEN komutunun, mutlaka değişken veya dizi oluşturulmasından önce kullanılması gerekir. Bunun sebebi, RS-232 kanalı açıldığında, CLR komutunun otomatik olarak işleme sokulmasıdır (bu işlem, belleğin baş kısmına yerleştirilen 512 bayta bağlıdır). OPEN yönergesi sırasında 512 baytlık boş yer olmaması halinde programınızın zarar göreceğini de aklınızdan çıkarmayın

RS-232 KANALINDAN VERİ ALMAK

RS-232 kanalından veri alırken, Commodore 64 alıcı deposu, taşmaya (overflow> neden olmadan 255 karakter bulundurabilir. Bu özellik RS-232 durum sözcüğünde belirtilir, (BASIC'te ST, makine dilinde RSSTAT). Taşma oluşursa, depo dolduktan sonra gelen tüm veriler kaybolur. Bu yüzden deponun mümkün olduğu kadar boş tutulması gerekir.

BASIC YAZIMI

ÖNERİLEN: GET# mdn, <yazınsal dizi değişkeni>
ÖNERİLMEYEN: INPUT# mdn, <değişken listesi>

KERNAL GİRİŞLERİ

CHKIN ($FFC6) - Giriş ve çıkış durumları hakkında daha fazla bilgi için bellek haritasına bakınız.

GETIN ($FFE4) - Giriş ve çıkış durumları hakkında daha fazla bilgi için bellek haritasına bakınız.

CHRIN ($FFCF) - Giriş ve çıkış durumları hakkında daha fazla bilgi için bellek haritasına bakınız.

NOTLAR:

Sözcük uzunluğu 8 bitten az ise, kullanılmayan tüm bitler 0 değerini alır.

GET# komutu kullanıldığında depoda hiçbir veri yoksa “” karakteri (boş dizi) elde edilir.

INPUT# kullanıldığında, sistem, boş olmayan bir karakter ve ardından satır başı alınmadıkça bekleme pozisyonundadır. Bu da gönderim için hazırım (CTS) veya teyp hazır (DSR) satırlarının INPUT# karakteri sırasında kaybolması halinde, sistemin RESTORE durumunda kalacağını belirtir. INPUT# ve CHRIN rutinlerinin tavsiye edilmemesinin nedeni budur.

RS-232 KANALINA VERİ GÖNDERİLMESİ

Veri gönderirken, deponun tümü dolmadan önce 255 karakter bulundurabileceğini biliyoruz. Bu durumda sistem, gönderime izin verilinceye kadar CHROUT rutininde bekleyecek veya **RUN/STOP** ve **RESTORE** tuşlarını kullanarak sistemin tekrar WARM START ile başlaması sağlanacaktır.

BASIC YAZIMI:

CMD mdn - BASIC'te belirtildiği gibi çalışır.
PRINT# mdn, <değişken listesi>

KERNAL GİRİŞLERİ:

CHKOUT ($FFC9) - Giriş ve çıkış durumları hakkında daha fazla bilgi için bellek haritasına bakınız.

CHROUT ($FFD2) - Giriş ve çıkış durumları hakkında daha fazla bilgi için bellek haritasına bakınız.

ÖNEMLİ NOTLAR: Çıkış kanalı için satır-sonu gecikmesi kullanılmaz. Bu da RS-232 yazıcısının, bir durma işlemi (Commodore 64'e bekleme için sorma) veya depolama olmadan doğru olarak yazamayacağı anlamındadır. Gecikme işlemi program içinde kolayca gerçekleştirilebilir Eğer CTS (x-hattı) haberleşmesi yapılıyorsa, RS-232 cihazı tarafından gönderime izin verilene kadar. Commodore 64 deposu dolacak ve bilgiyi burada tutacaktır. X haltı haberleşmesi veri alımı ve gönderimi için çok hat kullanan bir programdır.

CHKOUT programı, RS-232 arabirimleri için, EIA standartlarına uyarak x hattı haberleşmesini kullanır. RTS, CTS ve DCD hatları, Commodore 64'le birlikte veri terminali cihazı olarak tanımlanırlar.

RS·232 VERİ KANALININ KAPATILMASI

Bir RS-232 kanalını kapamak, çalışma anında depolardaki tüm verinin göz ardı edilmesi (gönderildiği veya yazıldığı önemli değildir), RS-232'nin veri alımını ve gönderimini durdurması, RTS'in işlenmesi, gönderilmiş veri (Sout) hattının 1 olması ve tüm RS-232 depolarının kaldırılması anlamındadır.

BASIC YAZIMI:

CLOSE mdn

KERNAL GİRİŞLERİ:

CLOSE ($FFC3) - Giriş ve çıkış durumları hakkında daha fazla bilgi için bellek haritasına bakınız.

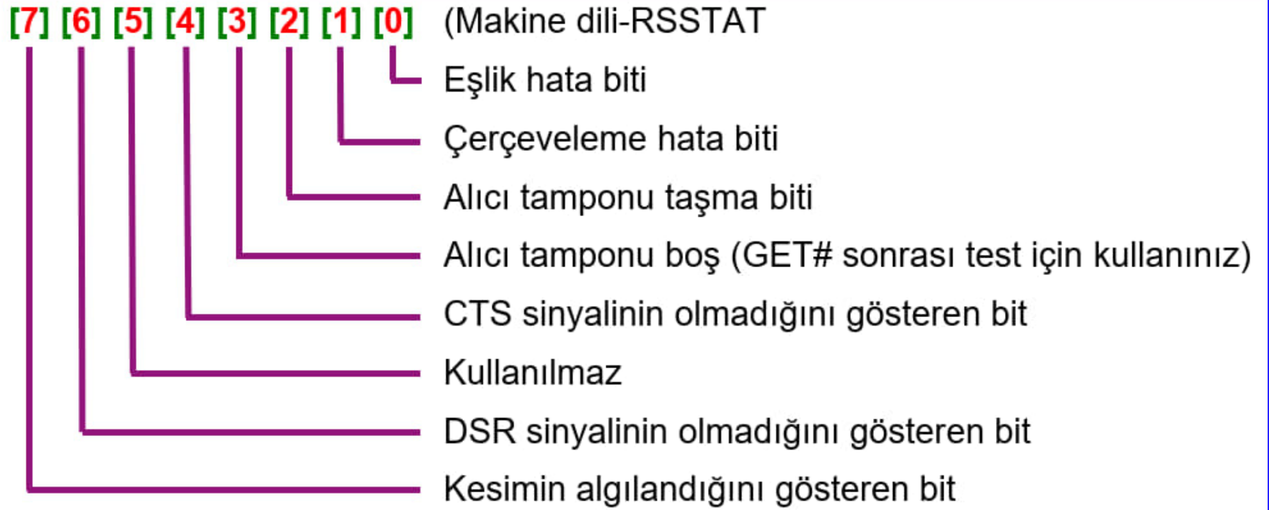
NOT: Kanalı kapatmadan önce, verinin tümünün gönderilmesine dikkat ediniz. BASIC'le bunu kontrol etmenin yolu şudur:

```
100 SS=ST: IF (SS=0 OR SS=8) THEN 100
110 CLOSE MDN
```

Tablo 6-1 Kullanıcı Portu Hatları

(6526 Cihaz 2 Yerleşimi ($DD00-$DD0F))						
UC	6526	TANIMLAMA	EIA	KISALTMA	GİRİŞ/ÇIKIŞ	MOD
C	PB0	Alınmış veri	(BB)	Sin	Giriş	1 2
D	PB1	Gönderim isteği	(CA)	RTS	Çıkış	1 *2
E	PB2	Veri terminali hazır	(CD)	DTR	Çıkış	1 *2
F	PB3	Halka göstergesi	(CE)	RI	Giriş	3
H	PB4	Alım hattı sinyali	(CF)	DCD	Giriş	2
J	PB5	Tanımlı değil	( )	XXX	Giriş	3
K	PB6	Gönderim için hazır	(CB)	CTS	Giriş	2
L	PB7	Veri kümesi hazır	(CC)	DSR	Giriş	2
B	FLAG2	Alınmış veri	(BB)	Sin	Giriş	1 2
M	PA2	Gönderilmiş veri	(BA)	Sout	Çıkış	1 2
A	GND	Koruyucu topraklama	(AA)	GND		1 2
N	GND	Sinyal topraklama	(AB)	GND		1 2 3

MODLAR:
1) 3-Hat arabirimi (Sin, Sout, GND)
2) X- Hat arabirimi
3) Sadece kullanıcı içindir. (Kullanılmaz/kod olarak işlenmez)
* Bu hatlar 3-Hat modunda, yüksek durumdadır.



Şekil 6-3 RS-232 Statü Kaydı

NOTLAR:

Eğer bit=0'sa hiç hata yoktur.

RS-232 statü kaydı BASIC'ten, ST değişkeni kullanılarak okunabilir.

Eğer ST, BASIC ya da KERNAL READST rutini ile okunduysa, programdan çıktığınızda RS-232 statü sözcüğü sıfırlanır. Statü sözcüğünün çok amaçlı kullanımı gerektiğinde, ST'nin bir diğer değişkene aktarılması gerekir.

Örneğin:

SR=ST: REM ST'Yİ, SR DEGISKENINE AKTAR

RS-232 statü sözcüğü sadece kullanılan son giriş/çıkış hattı RS-232 kanalı ise okunur (ve silinir).

ÖRNEK BASIC PROGRAMLARI

```
10 REM BU PROGRAM SESSIZ 700'E VERI GOND
ERIR VE ONDAN VERI ALIR
11 REM TERMINAL PET ASCII ICIN AYARLANDI
20 REM TI 700 SESSIZ KURULUM, 300 BAUD,
7-BIT ASCII, ISARETLEYICI PARITESI
21 REM TAM CIFT-YONLU
30 REM 3 HATLI ARAYUZ KULLANILAN BILGISA
YARDA AYNI KURULUM
100 OPEN 2,2,3,CHR$(6+32)+CHR$(32+128):R
EM KANAL AC
110 GET#2,A$:REM ALICI KANALINI AC
200 REM ANA DONGU
210 GET B$:REM BILGISAYARIN KLAVYESINDEN
AL
220 IF B$<>"" THEN PRINT#2,B$;:REM EGER
BIR TUSA BASIMIS ISE TERMINALE GONDER
230 GET#2,C$:REM TERMINALDEN BIR TUS AL
240 PRINT B$;C$;:REM BUTUN GIRISLERI BIL
GISAYAR EKRANINA BAS
250 SR=ST: IF SR=0 OR SR=8 THEN 200: REM
STATUS'U KONTROL ET IYI ISE DEVAM ET
300 REM HATA RAPORLARI
```

```
310 PRINT "HATA: ";
320 IF SR AND 1 THEN PRINT "ESLIKTE"
330 IF SR AND 2 THEN PRINT "CERCEVEDE"
340 IF SR AND 4 THEN PRINT "ALICI TAMPON
DOLU"
350 IF SR AND 128 THEN PRINT "BIRAK"
360 IF (PEEK(673) AND 1) THEN 360:REM TU
M KARAKTERLER GONDERILINCEYE KADAR BEKLE
370 CLOSE 2: END
```

```
10 REM BU PROGRAM GERCEK ASCII VERILERI
ALIR VE GONDERIR
100 OPEN 5,2,3,CHR$(6)
110 DIM F%(255),T%(255)
200 FOR J=32 TO 64:T%(J)=J:NEXT
210 T%(13)=13:T%(20)=8:RV=18:CT=0
220 FOR J=65 TO 90:K=J+32:T%=(J)=K:NEXT
230 FOR J=91 TO 95:T%(J)=J:NEXT
240 FOR J=193TO 218:K=J-128:T%(J)=K:NEXT
250 T%(146)=16:T%(133)=16
260 FOR J=0 TO 255
270 K=T%(J)
280 IF K<>0 THEN F%(K)=J:F%(K+128)=J
290 NEXT
300 PRINT" "CHR$(147)
310 GET#5,A$
320 IF A$=""OR ST<>0 THEN 360
330 PRINT" "CHR$(157);CHR$(F%(ASC(A$)));
340 IF F%(ASC(A$))=34 THEN POKE 212,0
350 GOTO 310
360 PRINT CHR$(RV)" "CHR$(157);CHR$(146)
;:GET A$
370 IF A$<>"" THEN
PRINT#5,CHR$(T%(ASC(A$)));
380 CT=CT+1
390 IF CT=8 THEN CT=0:RV=164-RV
410 GOTO 310
```

ALICI/VERİCİ DEPOSU TABAN ADRESİ GÖSTERGEÇLERİ

$00F7-RIBUF - Alıcı deponun taban adresini gösteren 2 baytlık göstergeç.
$00F9-ROBUF - Verici deponun taban adresini gösteren 2 baytlık göstergeç.

Yukarıdaki iki yerleşim. OPEN KERNAL rutini ile, herbiri 256 baytlık değişik bir depoyu gösterebilecek şekilde ayarlanır. Bunlar, üst baytları ($00F8 ve $00FA) 1 yapılarak yerlerini terk edebilirler (yok edilebilirler). Bu işlem de KERNAL CLOSE rutini ile yapılır. Makina dili programcısının bu yerleri kendi isteğine uygun olarak depo olarak kullanması (kaldırması) mümkündür.

Bu depoları yerleştirecek bir makina dili program kullanıldığında, belleğin başlangıcını belirten göstergeçleri doğru olarak kalıp kalmadığına dikkat etmek gerekir. Özellikle aynı zamanda BASIC programlarında da işletilmesi isteniyorsa .

SIFIRINCI–SAYFANIN BELLEKTEKİ YERLERİ VE RS-232 SİSTEM ARABİRİMİ İÇİN KULLANILMALARI

$00A7 – INBIT – Alıcı giriş biti için geçici saklama yeri
$00A8 – BITCI – Alıcı biti sayımı
$00A9 – RINONE – Alıcı bayrağı Başlangıç biti kontrolü
$00AA – RIDATA – Alıcı baytı depo/assembly yeri
$00AB – RIPRTY – Alıcı eşlik (parity) biti saklama yeri
$0084 – BITTS – Gönderici biti sayımı
$0085 – NXTBIT – Göndericinin yollayacağı bir sonraki bit
$00B6 – RODATA – Gönderici baytı depo/disassembly yeri

Yukarıdaki tüm sıfırıncı-sayfa yerleri lokal olarak kullanılırlar ve ilgili rutinlerin anlaşılması için bir rehber olarak verilmişlerdir. Bunlar RS-232 tipi işlemleri yapmak için BASIC ya da KERNAL programcıları tarafından doğrudan kullanılamazlar. RS-232 sistem rutinleri kullanılmalıdır.

0'INCI SAYFA OLMAYAN BELLEK YERLERİ VE RS-232 SİSTEM ARABİRİMİ İÇİN KULLANIMI

Genel RS-232 saklanımı:

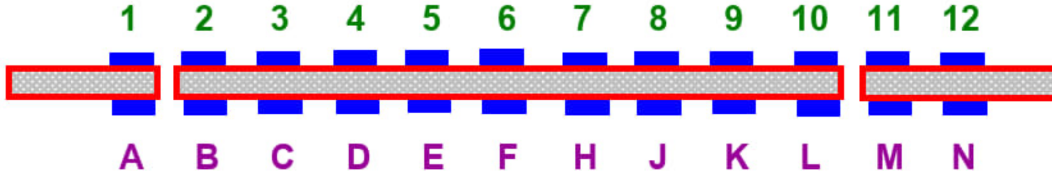
$0293 – M51CTR – Sahte 6551 kontrol kaydı (Şekil 6-1'e bakınız)
$0294 – M51COR – Sahte 6551 komut kaydı (Şekil 6-2'e bakınız)
$0295 – M51AJB – Dosya ismi kısmında, komut ve kontrol kaydından sonra gelen iki bayt. Bu yerleşimlerde, arabirimin çalışması sırasında, bit testi başlangıcı için baud hızı bulunur. Ve bu sonradan, baud hızının hesaplanmasında kullanılır.
$0297 – RSSTAT – AS-232 statü kaydı
$0298 – BITNUM – Gönderilecek/alınacak bit sayısı
$0299 – BAUDOF – Bir bit ünitesi zamanına eşit iki bayt (Sistem saati veya baud hızına bağlıdır).
$029B – RIDBE – Alıcı FIFO deponun sonu için bayt indeksi
$029C – RIDBS – Alıcı FIFO deponun başı için bayt indeksi
$029D – RODBS – Verici FIFO deponun başı için bayt indeksi
$029E – ROOBE – Verici FIFO deponun sonu için bayt indeksi
$02A1 – ENABL – CIA#2 ICR içinde, o anda aktif durumda olan kesintileri içerir. 4'üncü bitin çalışır durumda olması, sistemin alıcı kenar (receiver edge) için beklediği anlamına gelir. 1'nci bit çalışır durumda olduğunda sistem, veriyi alır. 0'ıncı bit çalışır durumda olduğunda da sistem, veriyi gönderir.

KULLANICI PORTU

Kullanıcı Portu, Commodore 64'ün dışarıyla bağlantısını sağlamak amacıyla yerleştirilmiştir. Bu porttaki hatları kullanarak Commodore 64'ü bir yazıcıya, MODEM' e hatta başka bir bilgisayara bağlayabilirsiniz.

Commodore 64'ün portu, doğrudan 6526CIA ciplerine bağlıdır. Programlayarak CIA'nın değişik cihazlara bağlanması sağlanır.

PORT UCU AÇIKLAMASI



PORT UCU AÇIKLAMASI

UÇ	İSİM	AÇIKLAMA
1	GND	Sistem toprağı
2	+5V	(100 mA MAX.)
3	RESET	Bu uçun topraklanması ile, Commodore 64'ün tam sıfırlanarak (COLD START) yapması sağlanır. BASIC programı için gerekli iki göstergeç sıfırlanır, fakat bellek temizlenmez. Bu aynı zamanda dış cihazlar için yeniden başlatma (RESET) çıkışıdır.
4	CNT1	CIA#1'den seri port sayacı (CIA ekine bakın)
5	SP1	CIA#1'den seri port (CIA 6526 özelliklerine bakınız.)
6	CNT2	CIA#2'den seri port sayacı (CIA ekine bakın)
7	SP2	CIA#1'den seri port (CIA 6526 özelliklerine bakınız.)
8	PC2	CIA#2'den haberleşme hattı
9	ATN	Bu uç, seri bağlantının ATN hattına bağlanır.
10	9 VAC +faz	Doğrudan Commodore 64 trafosuna bağlanır.
11	9 VAC -faz	(50 mA maksimum)
12	GND	Sistem toprağı
A	GND	Sistem toprağı
B	FLAG2	Commodore 64, CIA#1 çipindeki port B'nin kontrolünü sağlar. 2 hat, bir dış cihazla haberleşmek için, sekiz hat da giriş ve çıkış için kullanılır. B portundaki giriş/çıkış hatları, iki yerleşim kullanılarak kontrol edilir. Bir tanesi portun kendisi içindir ve 56577 ($DD01) adresinde yer alır. Normal olarak PEEK komutu ile giriş olarak okunabilir veya POKE komutu ile çıkış olarak tanımlanabilir. Veri yön kaydına uygun olan değerler yerleştirilerek, sekiz giriş/çıkış hattının her biri ya giriş ya da çıkış olarak tanımlanabilir.
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	Sistem toprağı

Veri yön kaydı, 56579 ($DD03) adresinde yer alır. Porttaki 8 hattın her birinin 8 bitlik veri yön kaydında (DDR) bir biti vardır ve bu bitin kontrolü ile hattın, giriş ya da çıkış amacıyla kullanılması sağlanır. DDR'deki bir bit 1 ise, karşılık gelen port, çıkış hattı, 0 ise bu port, giriş hattı olacaktır. Örneğin DDR'nin 3'üncü biti 1 ise, portun 3'ncü hattı çıkış olacaktır.

Başka bir örnek, DDR şu değeri aldıysa:

BİT NO : 7 6 5 4 3 2 1 0
DEĞER : 0 0 1 1 1 0 0 0

5,4,3'ncü hatlar çıkış, diğerleri ise giriş olacaktır.

Kullanıcı portuna bir değer yerleştirmek (POKE ile), ya da bu porttan bir değer okumak için (PEEK ile), hem veri yön kaydının hem de portun kendisinin kullanılması gerekir.

PEEK ve POKE yönergelerinde, kullanılacak sayının 0 ile 255 arasında olması gerekir. Örnekte verilen sayıların kullanılmadan önce, onlu sisteme çevrilmesi gerekir. Çevrildikten sonra değer:

$2^5 + 2^4 + 2^3 = 32 + 16 + 8 = 56$

DDR'deki herhangi bir bitin değerini bir yapmak için, 2'nin bit numarası kadar üssüne eşit olan sayıyı kullanmalısınız.

(16 = 2↑4 = 2*2*2*2, 8 = 2↑3 = 2*2*2)

Diğer iki hat, FLAG1 ve PA2, kullanıcı portundaki diğer hatlardan farklıdır. Bu iki hat, genelde el sıkışma (HANDSHAKE) türündeki haberleşme için kullanılır ve port B'den farklı bir şekilde programlanmıştır.

El sıkışma, iki cihazın haberleştiği zaman kullanılır. İki cihazdan biri, diğerinden hızlı çalışabileceğinden, birbirlerinin ne yaptığından haberdar olmaları gerekir. Cihazlar aynı hızla çalışsalar bile, bir cihazın, diğerinin veriyi ne zaman göndereceğini ve verinin alınıp alınmadığını bilebilmesi için, el sıkışma yönteminin kullanılması gerekir. FLAG1 hattının, el sıkışma yöntemi için çok uygun karakteristikleri vardır.

FLAG1, genel amaçlı kesinti girişi olarak kullanabilen, negatif kenarı algılayabilecek özellikte bir giriştir. FLAG hattındaki herhangi bir negatif geçiş, FLAG kesinti bitini 1 yapacaktır. Eğer FLAG kesintisi açılmışsa bu, bir kesinti isteğine neden olacaktır. Eğer FLAG biti açılmamışsa, program kontrolü altında, kesinti kaydından sırayla taranacaktır.

PA2, CIA'nın A portundaki ikinci bittir. Porttaki diğer bitler gibi kontrol edilir. Bu port 56576 ($DD00) adresine yerleştirilmiştir. Veri yön kaydı ise 56578 ($DD02) adresinde yer alır.

6526 hakkında daha detaylı bilgi için EK M'de bulunan 6526 çipinin özelliklerine bakiniz.

SERİ BAĞLANTI

Seri Bağlantı, Commodore 64'ün VIC 1541 disk sürücü ve VIC 1525 grafik çizici gibi cihazlarla haberleşmesini sağlayabilmek amacıyla, papatya zinciri (daisy chain) türünde tasarlanmıştır. Seri bağlantının avantajı, porta birden fazla cihazın bağlanabilmesidir. Seri bağlantıya aynı anda, 5 cihaz bağlanabilir.

Seri bağlantıda 3 çeşit işlem yapılır: Kontrol, konuşma ve dinleme. Kontrol cihazı, seri bağlantının çalışmasını kontrol eder. Konuşucu (talker), veriyi seri bağlantıya gönderir. Dinleyici (listener) ise, seri bağlantıdan veri alır.

Commodore 64 bağlantıyı kontrol eder. Ayrıca konuşucu, (örneğin yazıcıya bilgi gönderirken) ve dinleyici (disk sürücüden program yüklerken) gibi de davranabilir. Diğer cihazlar da dinleyici, konuşucu ya da her ikisi birden olabilirler. Fakat yalnızca, Commodore 64 kontrolör gibi davranabilir.

Seri bağlantıdaki tüm cihazlar, bağlantı üzerinden gönderilen tüm veriyi alacaktır. Commodore 64'ün göndereceği verinin, sadece gitmek istediği yer tarafından alınabilmesi için her cihazın bir bağlantı adresi vardır. Bu cihaz adresini kullanarak Commodore 64, bağlantıya erişimi kontrol eder. Seri bağlantı adresleri 4 ile 31 arasındaki herhangi bir değeri alabilirler.

Commodore 64, herhangi bir cihaza konuş ya da dinle komutu gönderebilir. Bir cihaza konuş komutu verildiğinde, cihaz seri bağlantıya verileri yerleştirmeye başlar. Dinle komutundan sonra ise cihaz verileri almak için hazır duruma geçer (Commodore 64'ten veya seri bağlantıdaki herhangi bir cihazdan). Bağlantı üzerindeki cihazlardan, bir kerede yalnızca bir tanesi konuşabilir. Aksi takdirde veriler karışacak ve sistem bu karışıklıktan ötürü duracaktır. Buna karşılık, bir cihaz konuşurken, aynı anda birden fazla cihaz dinleyebilir.

SERİ VERİ YOLU ADRESLERİ

CİHAZ NUMARASI	CİHAZ
4	Commodore yazıcı
5	Commodore yazıcı
8	VIC-1541 disk sürücü
9	VIC-1541 disk sürücü
10	VIC-1541 disk sürücü
11	VIC-1541 disk sürücü

Diğer cihaz adresleri de kullanılabilir. Her cihazın kendi adresi vardır. Bazı cihazlar (Commodore 64 yazıcısı gibi), kullanıcının isteğine göre iki adresten birini alabilirler. İkincil adres, Commodore 64'ün bir cihaza çalışma durumu bilgisi göndermesi amacıyla kullanılır. Örneğin hem kanalı açmak hem de bağlantıdaki yazıcının büyük/küçük harf modunda çalışmasını sağlamak için şu yönergeyi kullanmalısınız:

```
OPEN 1,4,7
```

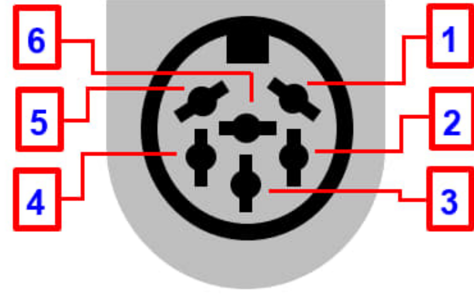
burada.

1 mantıksal dosya numarası
4 yazıcının cihaz numarası adresi, ve
7 yazıcının büyük/küçük harf modunda çalışmasını sağlayan ikincil adres

Seri bağlantının çalışmasında 3'ü giriş, 3'ü de çıkış olmak üzere 6 hat kullanılır. 3 giriş hattı Commodore 64'e veri, kontrol ve zamanlama sinyalleri getirir. 3 çıkış hattı ise Commodore 64'ten diğer cihazlara veri, kontrol ve zamanlama sinyalleri götürür.

SERİ BAĞLANTI ÇIKIŞ UÇLARI

UÇ	TANIMLAMA
1	Seri SRQ içeri
2	GND
3	Seri ATN içeri/dışarı
4	Seri CLK içeri/dışarı
5	Seri veri içeri/dışarı
6	Bağlantı yok



SERİ SRQ İÇERİ (İÇERİ SERİ SERVİS İSTEĞİ)

Seri bağlantıdaki herhangi bir cihaz, Commodore 64'ün kendisi ile ilgilenmesini istediğinde bu sinyali alçak düzey (LOW) haline getirebilir. Bunun sonucunda, Commodore 64 bu cihazla çalışmaya başlar (Şekil 6-4 ·e bakınız).

İLGİLENİLEN CİHAZLARA BAYT GÖNDERİMİ
Dinleyici = Alıcı, Konuşmacı = Verici)
NORMAL VERİ BAYTI
ATN
SAAT
VERİ
T_{S}
T_{AT}
T_{NE}
T_{V}
T_{R}
0 1 2 3 4 5 6 7
LSB
MSB
T_{H}
T_{F}
VERİ GEÇERLİ
DİNLEYİCİ VERİ İÇİN HAZIR
DİNLEYİCİ VERİYİ KABUL ETTİ

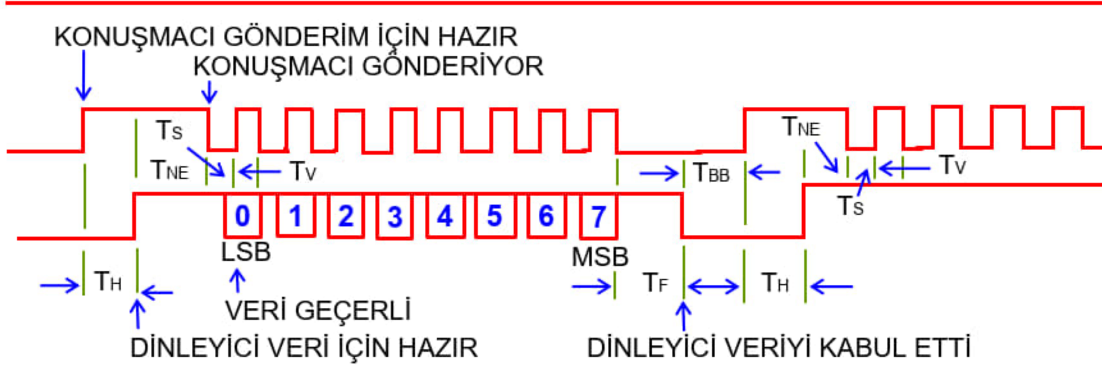
HABERLEŞİM SONU VEYA TANIMLANMASI (Mesajın son baytı)
ATN
KONUŞMACI GÖNDERİM İÇİN HAZIR
KONUŞMACI GÖNDERİYOR
SAAT
T_{BB}
T_{S}
T_{V}
VERİ
4 5 6 7
MSB
T_{H}
T_{YE}
T_{EI}
T_{RY}
T_{F}
T_{FR}
DİNLEYİCİ VERİ İÇİN HAZIR
EOI-EL SIKIŞMA ZAMAN AŞIMI
SİSTEM HATTI ÇIKIŞI
DİNLEYİCİ HAZIR

KONUŞMA DİKKAT ÇEKİMİ TERS DÖNER (KONUŞMACI VE DİNLEYİCİ DEĞİŞİR)
ATN
CİHAZ ŞİMDİ SIRANIN KONUŞMACIDA OLDUĞUNU BİLDİRİR
KONUŞMACI GÖNDERİM İÇİN HAZIR
SAAT
T_{NE}
T_{R}
T_{DC}
T_{DA}
T_{V}
T_{TK}
VERİ
4 5 6 7
T_{S}
0 1 2 3 4 5 6 7
MSB
T_{F}
T_{H}
LSB
MSB
T_{F}
DİNLEYİCİ OLUR. SAAT = YÜKSEK, VERİ DÜŞÜK
VERİ İÇİN HAZIR

Şekil 6-4 Seri Veri Yolu

SERİ ATN İÇERİ/DIŞARI (İÇERİ/DIŞARI DİKKAT ÇEKME)

Commodore 64, bu sinyali seri bağlantıdaki herhangi bir cihaz için komut dizisine başlamak amacıyla kullanır. Commodore 64 bu sinyali alçak seviyeye getirirse, bağlantıdaki tüm cihazlar Commodore 64'ün göndereceği adres için dinleme durumuna geçer. Adresi gönderilen cihazın belirli bir sure içerisinde karşılık vermesi gerekir, aksi takdirde Commodore 64, cihazın bağlantıda olmadığını düşünüp STATÜ SÖZCÜĞÜ'nde hata uyarısı verecektir (Şekil 6-4·e bakınız).



SERİ VERİ YOLU ZAMANLAMASI

TANIMLAMA	SEMBOL	MIN.	TYP.	MAX.
ATN karşılığı (mutlaka gerekli)[1]	T_{AT}	-	-	1000μS
Dinleyici beklemede	T_H	0	-	∞
RFD'ye EOI olmayan karşılık[2]	T_{NE}	-	40μS	200μS
Bit yükleme konuşucu	T_S	20μS	70μS	-
Geçerli veri	T_V	20μS	20μS	-
Çerçeve el sıkışması[3]	T_F	0	20	1000μS
ATN'nin bırakılması için çerçeve	T_R	20μS	-	-
Baytlar arası zaman	T_{BB}	100μS	-	-
EOI karşılığı zamanı	T_{YE}	200μS	250μS	-
EOI karşılığı tutma zamanı[5]	T_{EI}	60μS	-	-
Verici karşılığı sınırı	T_{RY}	0	30μS	60μS
Onaylama baytı[4]	T_{PR}	20μS	30μS	-
Dikkat çekme sonu konuşması	T_{TK}	20μS	30μS	100μS
Dikkat çekme onayı konuşması	T_{DC}	0	-	-
Dikkat çekme onayı tutma zamanı	T_{DA}	60μS	-	-
EOI onayı	T_{FR}	80μS	-	-

NOT:
1. Maksimum zaman aşılırsa, cihaz yok hatası
2. Maksimum zaman aşılırsa, EOI karşılığı gerekir
3. Maksimum zaman aşılırsa, çerçeve hatası
4. Dışardaki cihazın konuşmacı olabilmesi için, T ve T_{PR} minimumunun 60μS olması gerekir.
5. Dışardaki cihazın dinleyici olabilmesi için, T_{EI} minimumunun 80μS olması gerekir.

İÇERİ/DIŞARI SERİ CLK (CLK = CLOCK: Saat)

Bu sinyal seri bağlantıda gönderilen verinin zamanlaması için kullanılır (Şekil 6-4'e bakınız).

İÇERİ/DIŞARI SERİ VERİ

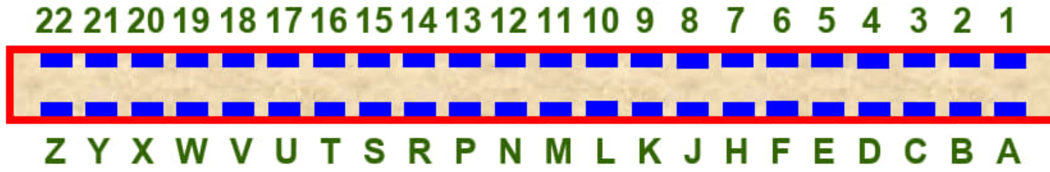
Seri bağlantı üzerindeki veri, bu hatta bir kerede bir bit olarak gönderilir (Şekil 6-4'e bakınız).

GENİŞLEME PORTU

Genişleme bağlayıcısı, Commodore 64'ün arka kısmında yer alan, dişi ve 44 uçludur (22/22). Commodore 64 size doğru olmak üzere, gelişim bağlayıcısı bilgisayarınızın en arka sağ köşesindedir. Kullanılabilmesi için gene 44 uçlu (22/22) erkek bağlayıcıya ihtiyaç vardır.

Bu port, Commodore 64 sisteminin, veri ve adres veri yollarına dışardan erişebilmesi amacıyla kullanılır. Genişleme veri yolunu kullanırken, cihazınızı yanlış bağlarsanız Commodore 64'e zarar verebilirsiniz. Bu yüzden çok dikkatli olun.

Genişleme veri yolu şu şekilde düzenlenmiştir.



Bağlayıcıdaki sinyaller şunlar olacaktır.

UÇ	İSİM	AÇIKLAMA
1	GND	Sistem toprağı
2	+5VDC	(Tüm kullanıcı portu ve kartuş cihazları 450 mA'den fazla akım çekemezler.
3	+5VDC	
4	IRQ	6502'ye kesinti isteği hattı (aktif alçak)
5	R/W	Okuma/yazma (yazma aktif alçak)
6	DOT CLOCK	8.18 MHz video nokta saati
7	I/O1	Giriş/çıkış bloğu 1 @ $DE00-$DEFF (aktif alçak) depolanmamış giriş/çıkış
8	GAME	ls ttl girişi (aktif alçak)
9	EXROM	ls ttl girişi (aktif alçak)
10	I/O2	Giriş/çıkış bloğu 2 @ $DF00-$DFFF (aktif alçak) depolanmış ls ttl çıkışı
11	ROML	8K çözülü RAM/ROM bloğu @ $8000 (aktif alçak) depolanmış ls ttl çıkışı
12	BA	VIC-II çipinden veri yolu hazır sinyali, depolanmamış maksimum 1 ls ttl yüklenmesi
13	DMA	Doğrudan bellek erişimi isteği sinyali (aktif alçak) girişi ls ttl girişi

UÇ	İSİM	AÇIKLAMA	
14	D7	Veri yolu biti 7	
15	D6	Veri yolu biti 6	
16	D5	Veri yolu biti 5	
17	D4	Veri yolu biti 4	Depolanmamış maksimum
18	D3	Veri yolu biti 3	1 ls ttl yüklenmesi
19	D2	Veri yolu biti 2	
20	D1	Veri yolu biti 1	
21	D0	Veri yolu biti 0	
22	GND	Sistem toprağı	
A	GND	Sistem toprağı	
B	ROMH	8K çözülü RAM/ROM bloğu @ $E000 depolanmış	
C	RESET	6502 reset ucu depolanmış çıkış/depolanmamış giriş	
D	NMI	6502 önlenemeyen kesim depolanmış Çı/depolanmamış Gi	
E	•2	Faz 2 sistem saati	
F	A15	Adres veri yolu biti 15	
H	A14	Adres veri yolu biti 14	
J	A13	Adres veri yolu biti 13	
K	A12	Adres veri yolu biti 12	
L	A11	Adres veri yolu biti 11	
M	A10	Adres veri yolu biti 10	
N	A9	Adres veri yolu biti 9	
P	A8	Adres veri yolu biti 8	Depolanmamış maksimum
R	A7	Adres veri yolu biti 7	1 ls ttl yüklenmesi
S	A6	Adres veri yolu biti 6	
T	A5	Adres veri yolu biti 5	
U	A4	Adres veri yolu biti 4	
V	A3	Adres veri yolu biti 3	
W	A2	Adres veri yolu biti 2	
X	A1	Adres veri yolu biti 1	
Y	A0	Adres veri yolu biti 0	
Z	GND	Sistem toprağı	

Not: Üstteki çizgiler (.......... gibi) aktif alçak anlamına gelir.

Aşağıda genişleme portunda yer alan bazı önemli hatların açıklaması verilmiştir:

Uç 1, 22, A ve Z sistem toprağına bağlıdır.

Uç 6 nokta saatidir. Tüm sistem zamanlamasının elde edildiği bu birim 8.18 MHz video nokta saati olarak görev yapar.

Uç 12 VIC-II çipinden gelen BA (Bus available: Veri yolu hazır) sinyalini alır. Bu hat, VIC-II sistem veri yolunu almadan 3 çevrim önce alçak olur ve görüntü bilgilerinin alınması bitinceye kadar bu durumda kalır.

Uç 13 DMA (Doğrudan Bellek Erişimi) hattıdır. Bu hat alçak olduğunda, 6510 işlemci çipinin adres ve veri yolu ile okuma/yazma hattı yüksek direnç durumu moduna girer. Bu hattın, sadece •2 saati alçak olduğu zaman, alçak haline getirilmesi gerekir. Ayrıca, VIC-II çipi, DMA görüntüleme işlemine devam edeceğinden dışardaki cihazın VIC-II zamanlamasına uyması gerekir. (VIC-II zamanlamasına bakınız.) Bu hat Commodore 64'te yüksek seviyeye getirilir.

Z-80 MİKROİŞLEMCİ KARTUŞU

Bu kitabı okumak ve bilgisayarınızı kullanmak size Commodore 64'ün gerçekte ne kadar çok yönlü olduğunu gösterdi. Ancak bu makineyi ihtiyaçlarınızı karşılama konusunda daha da yetenekli kılan şey, çevre birimlerinin eklenmesidir. Çevre birimleri Datasette™ kaydediciler, disk sürücüleri, yazıcılar ve modemler gibi şeylerdir. Tüm bu öğeler Commodore 64'ünüze makinenizin arkasındaki çeşitli bağlantı noktaları ve soketler aracılığıyla eklenebilir. Commodore çevre birimlerini bu kadar iyi yapan şey, çevre birimlerimizin "akıllı" olmasıdır. Bu, kullanımdayken değerli Rasgele Erişim Belleği alanını kaplamadıkları anlamına gelir. Commodore 64'ünüzde 64K belleğin tamamını kullanmakta özgürsünüz. Commodore 64'ünüzün bir başka avantajı da Commodore 64'ünüzde bugün yazdığınız programların çoğunun gelecekte satın alacağınız herhangi bir yeni Commodore bilgisayarla yukarı doğru uyumlu olacağı gerçeğidir. Bunun nedeni kısmen bilgisayarın İşletim Sisteminin (OS) nitelikleridir. Ancak Commodore işletim sisteminin yapamayacağı bir şey var: programlarınızı başka bir şirket tarafından üretilen bir bilgisayarla uyumlu hale getirmek.

Çoğu zaman başka bir şirketin bilgisayarını kullanmayı düşünmenize bile gerek kalmayacak çünkü Commodore 64'ün kullanımı çok kolay. Ancak Commodore 64 formatında bulunmayan yazılımlardan yararlanmak isteyen ara sıra kullanıcılar için bir Commodore CP/M® kartuşu oluşturduk.

CP/M® "bilgisayara bağımlı" bir işletim sistemi değildir. Bunun yerine, normalde programlama için mevcut olan bellek alanının bir kısmını kendi işletim sistemini çalıştırmak için kullanır. Bunun avantajları ve dezavantajları var. Dezavantajları ise yazdığınız programların Commodore 64'ün yerleşik işletim sistemini kullanarak yazabileceğiniz programlardan daha kısa olması gerekmesidir. Ayrıca Commodore 64'ün güçlü ekran düzenleme yeteneklerini KULLANAMAZSINIZ. Avantajları, artık CP/M® ve Z-80 mikroişlemci için özel olarak tasarlanmış çok sayıda yazılımı kullanabilmeniz ve CP/M® işletim sistemini kullanarak yazdığınız programların taşınabilmesi ve üzerinde çalıştırılabilmesidir. CP/M® ve Z-80 kartı olan başka herhangi bir bilgisayar.

CP/M® "bilgisayara bağımlı" bir işletim sistemi değildir. Bunun yerine, normalde programlama için mevcut olan bellek alanının bir kısmını kendi işletim sistemini çalıştırmak için kullanır. Bunun avantajları ve dezavantajları var. Dezavantajları ise yazdığınız programların Commodore 64'ün yerleşik işletim sistemini kullanarak yazabileceğiniz programlardan daha kısa olması gerekmesidir. Ayrıca Commodore 64'ün güçlü ekran düzenleme yeteneklerini KULLANAMAZSINIZ. Avantajları, artık CP/M® ve Z-80 mikroişlemci için özel olarak tasarlanmış çok sayıda yazılımı kullanabilmeniz ve CP/M® işletim sistemini kullanarak yazdığınız programların taşınabilmesi ve üzerinde çalıştırılabilmesidir. CP/M® ve Z-80 kartı olan başka herhangi bir bilgisayar.

COMMODORE CP/M® KULLANIMI

Commodore Z-80 kartuşu, Commodore 64'ünüzde Z-80 mikroişlemci için tasarlanmış programları çalıştırmanıza olanak tanır. Kartuş, Commodore CP/M® işletim sistemini içeren bir disketle birlikte sağlanır.

COMMODORE CP/M®'İ ÇALIŞTIRMAK

CP/M®'yi çalıştırmak için:

1. CP/M® programını disk sürücünüzden YÜKLEYİN.
2. ÇALIŞTIR yazın.
3. **RETURN** tuşuna basın.

Bu noktada Commodore 64'teki 64K bayt RAM'e yerleşik 6510 merkezi işlemci tarafından erişilebilir veya Z-80 merkezi işlemci için 48K bayt RAM mevcuttur. Bu iki işlemci arasında ileri geri geçiş yapabilirsiniz ancak bunları aynı anda tek bir programda KULLANAMAZSINIZ. Bu, Commodore 64'ünüzün gelişmiş zamanlama mekanizması sayesinde mümkündür.

Aşağıda Z-80 kartuşunda gerçekleştirilen bellek adresi çevirisi yer almaktadır. CP/M® $1000 (hex)'de kullanılan bellek konumlarına 4096 bayt ekleyerek normal Commodore 64 işletim sisteminin bellek adreslerine eşit olduğunuzu fark edeceksiniz. Z-80 ve 6510 hafıza adresleri arasındaki yazışmalar şu şekildedir:

Z-80 ADRESLERİ		6510 ADRESLERİ	
ONLUK	ONALTILIK	ONLUK	ONALTILIK
0000–4095	0000–0FFF	4096–8191	1000–1FFF
4096–8191	1000–1FFF	8192–12287	2000–2FFF
8192–12287	2000–2FFF	12288–16383	3000–3FFF
12288–16383	3000–3FFF	16384–20479	4000–4FFF
16384–20479	4000–4FFF	20480–24575	5000–5FFF
20480–24575	5000–5FFF	24576–28671	6000–6FFF
24576–28671	6000–6FFF	28672–32767	7000–7FFF
28672–32767	7000–7FFF	32768–36863	8000–SFFF
32768–36863	8000–8FFF	36864–40959	9000–9FFF
36864–40959	9000–9FFF	40960–45055	A000–AFFF
40960–45055	A000–AFFF	45056–49151	B000–BFFF
45056–49151	B000–BFFF	49152–53247	C000–CFFF
49152–53247	C000–CFFF	53248–57343	D000–DFFF
53248–57343	D000–DFFF	57344–61439	E000–EFFF
57344–61439	E000–EFFF	61440–65535	F000–FFFF
61440–65535	F000–FFFF	0000–4095	0000–0FFF

Z-80'i AÇMAK ve 6510 yongasını KAPATMAK için aşağıdaki örnek programı yazın ve çalıştırın.

Commodore CP/M® ve Z-80 mikroişlemci hakkında daha fazla ayrıntı için yerel Commodore bilgisayar satıcınızda kartuşu ve Z-80 Referans Kılavuzunu arayın.

```
10 REM BU PROGRAM Z80 KART ILE KULLANILI
R
20 REM ONCE Z80 VERILERINI $1000 (Z80=$
0000)'DA DEPOLAR,
30 REM SONRA 6510 IRQ'LARI KAPATIR VE
VE Z80 KARTINI ETKINLESTIRIR.
40 REM 6510 SISTEMININ YENIDEN ETKINLEST
IRILMESI ICIN
50 REM Z80 KARTININ KAPATILMASI GEREKLID
IR.
100 REM Z80 VERILERI DEPOLAYIN
110 READ B: REM TASINACAK Z80 KODUNUN BO
YUTUNU ALIN
120 FOR I=4096 TO 4096+B-1:REM KODU TASI
130 READ A:POKE I,A
140 NEXT I
200 REM Z80 KODU CALISTIR.
210 POKE 56333,127: REM 6510 IRQ'LARI KA
PATIN
220 POKE 56832,00 : REM Z80 KARTI ACIN
230 POKE 56333,129: REM 6510 IRQ'LARI, Z
80'I TAMAMLADIKTAN SONRA ACIN
240 END
1000 REM Z80 MAKINE DILI KODU VERI BOLUM
U
1010 DATA 18 : REM AKTARILACAK VERI BOYU
TU
1100 REM Z80 ACMA KODU
1110 DATA 00,00,00 : REM Z80 KARTIMIZ AC
ILMA ZAMANI GEREKLIDIR $0000'DA
1200 REM Z80 GOREV VERILERI BURADA
1210 DATA 33,02,245: REM LD HL,NN (EKRAN
DAKI KONUM)
1220 DATA 52 : REM INC HL (KONUMU ARTIRI
N)
1300 REM Z80 KENDI KENDINI KAPANMA VERI
BURADA
1310 DATA 62,01 : REM LD A,N
1320 DATA 50,00,206 : REM LD (NN),A :G/C
KONUMU
1330 DATA 00,00,00 : REM YOK, YOK, YOK
1340 DATA 195,00,00 : REM JMP $0000
```

EKLER

EK A
BASIC SÖZCÜKLERİ İÇİN KISALTMALAR

Commodore 64 zamandan tasarruf edebilmeniz için, birçok BASIC sözcüğünü kısaltılmış ifadeleri ile kabul edebilecek gibi tasarlanmıştır. Örneğin PRINT sözcüğünün kısaltılmışı, bir soru (?) işaretidir. Diğer sözcüklerin kısaltmaları, o sözcüğün ilk bir ya da ikinci harfi ile SHIFT'le basılan bir sonraki harften oluşur. Kısaltmalarla yazılan programların dökümleri çıkarıldığında bilgisayar sözcükleri olmaları gerektiği biçimlerde yazacaktır. Bazı kısaltmalarda sol parantezin kullanılmış olduğuna dikkatinizi çekeriz.

KOMUT	KISALTMA	EKRAN	KOMUT	KISALTMA	EKRAN
ABS	A SHIFT + B	A▕	INPUT#	I SHIFT + N	I╱
AND	A SHIFT + N	A╱	INT	INT	INT
ASC	A SHIFT + S	A♥	LEFT$	LE SHIFT + F	LE▂
ATN	A SHIFT + T	A▎	LEN	LEN	LEN
CHR$	C SHIFT + H	C▕	LET	L SHIFT + E	L▔
CLOSE	CL SHIFT + O	CL┌	LIST	L SHIFT + I	L╮
CLR	C SHIFT + L	C└	LOAD	L SHIFT + O	L┌
CMD	C SHIFT + M	C╲	LOG	LOG	LOG
CONT	C SHIFT + O	C┌	MID$	M SHIFT + I	M╮
COS	COS	COS	NEW	NEW	NEW
DATA	D SHIFT + A	D♠	NEXT	N SHIFT + E	N▔
DEF	D SHIFT + E	D▔	NOT	N SHIFT + O	N┌
DIM	D SHIFT + I	D╮	ON	ON	ON
END	E SHIFT + N	E╱	OPEN	O SHIFT + P	O┐
EXP	E SHIFT + X	E♣	OR	OR	OR
FN	FN	FN	PEEK	P SHIFT + E	P▔
FOR	F SHIFT + O	F┌	POKE	P SHIFT + O	P┌
FRE	F SHIFT + R	F▁	POS	POS	POS
GET	G SHIFT + E	G▔	PRINT	?	?
GET#	GET#	GET#	PRINT#	P SHIFT + R	P▁
GOSUB	GO SHIFT + S	GO♥	READ	R SHIFT + E	R▔
GOTO	G SHIFT + O	G┌	REM	REM	REM
IF	IF	IF	RESTORE	RE SHIFT + S	RE♥
INPUT	INPUT	INPUT	RETURN	RE SHIFT + T	RE▎

KOMUT	KISALTMA	EKRAN	KOMUT	KISALTMA	EKRAN
RIGHT$	R SHIFT + I	R╮	SYS	S SHIFT + Y	S│
RND	R SHIFT + N	R╱	TAB	T SHIFT + A	T♠
RUN	R SHIFT + U	R╭	TAN	TAN	TAN
SAVE	S SHIFT + A	S♠	THEN	T SHIFT + H	T│
SGN	S SHIFT + G	S│	TIME	TI	TI
SIN	S SHIFT + I	S╮	TIME$	TI$	TI$
SPC	S SHIFT + P	S┐	TO	TO	TO
SQR	S SHIFT + Q	S●	USR	U SHIFT + S	U♥
STATUS	ST	ST	VAL	V SHIFT + A	V♠
STEP	ST SHIFT + E	ST▔	VERIFY	V SHIFT + E	V▔
STOP	S SHIFT + T	S│	WAIT	W SHIFT + A	W♠
STR$	ST SHIFT + R	ST▁			

EK B
EKRAN GÖSTERİM KODLARI

İzleyeceğiniz tabloda Commodore 64'ün bütün karakterleri gösterilmiştir. İstenilen bir karakter için, denk gelen ekran bellek adresine hangi sayının yerleştirilmesi gerektiğini (POKE ile) bu tablodan çıkarabilirsiniz. Aynı zamanda PEEK ile hangi sayıya hangi karakterin karşılık düştüğünü yine bu tablodan çıkarabilirsiniz.

Bildiğiniz gibi Commodore 64'ün iki karakter takımı vardır. Yani birinci takımdaki karakterleri ekranda yazarken ikinci takımı kullanmazsınız. Takımlar arası geçiş SHIFT ve C= (Commodore) tuşları ile gerçekleştirilir. BASIC dilinde, POKE 53272,21 büyük harf moduna, POKE 53272,23 ise küçük harf moduna geçişi verecektir. Tabloda bulunan herhangi bir sayı, aynı zamanda negatif gösterim modunda elde edilebilir. Negatif gösterim kodu, her karakterin kendi koduna 128 eklenerek bulunacaktır.

1504'üncü ekran hanesinde bir top görmek istiyorsanız bu karakterin kodunu (81), 1504'üncü ekran bellek adresine yerleştireceksiniz: POKE 1504,81.

Her karakterin rengini kontrol eden bir bellek grubu daha vardır. 55296-56295 adresleri. Deminki dolu daireyi sarıya dönüştürmek için (renk kodu 7) gerekli bellek birimine 7 yerleştireceksiniz: POKE 55776, 7.

Ek D'de ekran ve bellek haritaları ve renk kodları verilmiştir.

EKRAN KODLARI

SET1	SET2	POKE	SET1	SET2	POKE	SET1	SET2	POKE
@		0	Q	q	17	"		34
A	a	1	R	r	18	#		35
B	b	2	S	s	19	$		36
C	c	3	T	t	20	%		37
D	d	4	U	u	21	&		38
E	e	5	V	v	22	'		39
F	f	6	W	w	23	(		40
G	g	7	X	x	24	)		41
H	h	8	Y	y	25	*		42
I	i	9	Z	z	26	+		43
J	j	10	[		27	,		44
K	k	11	£		28	-		45
L	l	12	]		29	.		46
M	m	13	↑		30	/		47
N	n	14	←		31	0		48
O	o	15	SPACE		32	1		49
P	p	16	!		33	2		50

SET1	SET2	POKE	SET1	SET2	POKE	SET1	SET2	POKE
3		51		M	77			103
4		52		N	78			104
5		53		O	79			105
6		54		P	80			106
7		55		Q	81			107
8		56		R	82			108
9		57		S	83			109
:		58		T	84			110
;		59		U	85			111
<		60		V	86			112
=		61		W	87			113
>		62		X	88			114
?		63		Y	89			115
		64		Z	90			116
	A	65			91			117
	B	66			92			118
	C	67			93			119
	D	68			94			120
	E	69			95			121
	F	70	SPACE		96			122
	G	71			97			123
	H	72			98			124
	I	73			99			125
	J	74			100			126
	K	75			101			127
	L	76			102			

126-255 arası kodlar 0-127 arası kodların negatif halleridir.

NOT: Aşağıdaki POKE'lar set 1 ve 2'de aynı sembolü görüntüler: 1, 27 ila 64 arası, 91 ila 93 arası, 96 ila 104 arası, 106 ila 121 arası, 123 ila 127 arası.

EK C
ASCII VE CHR$ KODLARI

Bu ek, X'in her olası değeri için PRINT CHR$(X)'in görüntüleyeceği karakterleri vermektedir. Aynı zamanda X'in karakter olması halinde PRINT ASC("X") yazılarak elde edilecek değerler de sıralanmıştır. GET komutuyla alınan karakterlerin değerlendirilmesi, küçük büyük harf çevrimleri veya karakter bazlı komutlarda bu liste faydalı olacaktır.

BASILAN	CHR$	BASILAN	CHR$	BASILAN	CHR$	BASILAN	CHR$
	0		24	/	47	F	70
	2		25	0	48	G	71
	3		26	1	49	H	72
	4		27	2	50	I	73
WHT	5	RED	28	3	51	J	74
	6	CRSR→	29	4	52	K	75
	7	GRN	30	5	53	L	76
	8	BLU	31	6	54	M	77
SHIFT C= Pasif	9	SPACE	32	7	55	N	78
SHIFT C= Aktif	10	!	33	8	56	O	79
	11	"	34	9	57	P	80
	12	#	35	:	58	Q	81
RETURN	13	$	36	;	59	R	82
Küçük harfe geçiş	14	%	37	<	60	S	83
	15	&	38	=	61	T	84
	16	'	39	>	62	U	85
CRSR↓	17	(	40	?	63	V	86
RVS ON	18	)	41	@	64	W	87
CLR HOME	19	*	42	A	65	X	88
INST DEL	20	+	43	B	66	Y	89
	21	,	44	C	67	Z	90
	22	-	45	D	68	[	91
	23	.	46	E	69	£	92

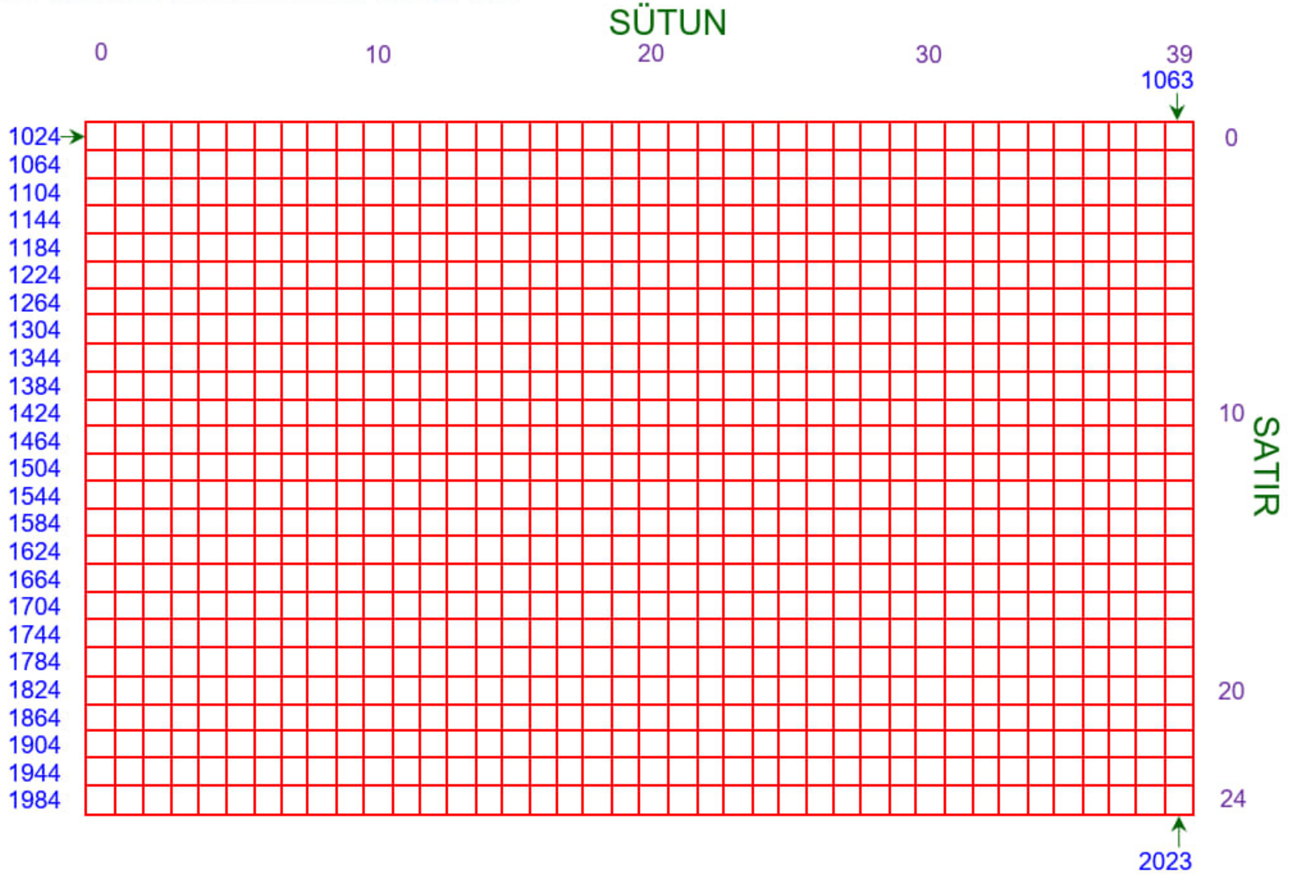
BASILAN	CHR$	BASILAN	CHR$	BASILAN	CHR$	BASILAN	CHR$
]	93		118		143		168
↑	94		119	BLK	144		169
←	95		120	CRSR ↑	145		170
	96		121	RVS OFF	146		171
	97		122	CLR HOME	147		172
	98		123	INST DEL	148		173
	99		124	Kahverengi	149		174
	100		125	Açık kırmızı	150		175
	101		126	Gri 1	151		176
	102		127	Gri 2	152		177
	103		128	Açık yeşil	153		178
	104	Turuncu	129	Açık mavi	154		179
	105		130	Gri 3	155		180
	106		131	PUR	156		181
	107		132	←CRSR	157		182
	108	F1	133	YEL	158		183
	109	F2	134	CYN	159		184
	110	F3	135	SPACE	160		185
	111	F4	136		161		186
	112	F5	137		162		187
	113	F6	138		163		188
	114	F7	139		164		189
	115	F8	140		165		190
	116	SHIFT RETURN	141		166		191
	117	Büyük harfe geçiş	142		167		

192-223 arası kodlar 96-127 arası kodlar ile aynı
224-254 arası kodlar 160-190 arası kodlar ile aynı
255 kodu 126 kodu ile aynı

EK D
EKRAN VE RENK BELLEK HARİTALARI

Aşağıdaki tablolarda, karakterlerin ekran üzerindeki konumlarının, hangi bellek yerleşimleri tarafından kontrol edildiği ve karakterlerin renk kodları ile, karakterlerin rengini değiştirmek için kullanılan yerleşimler yer almaktadır.

EKRAN BELLEK HARİTASI

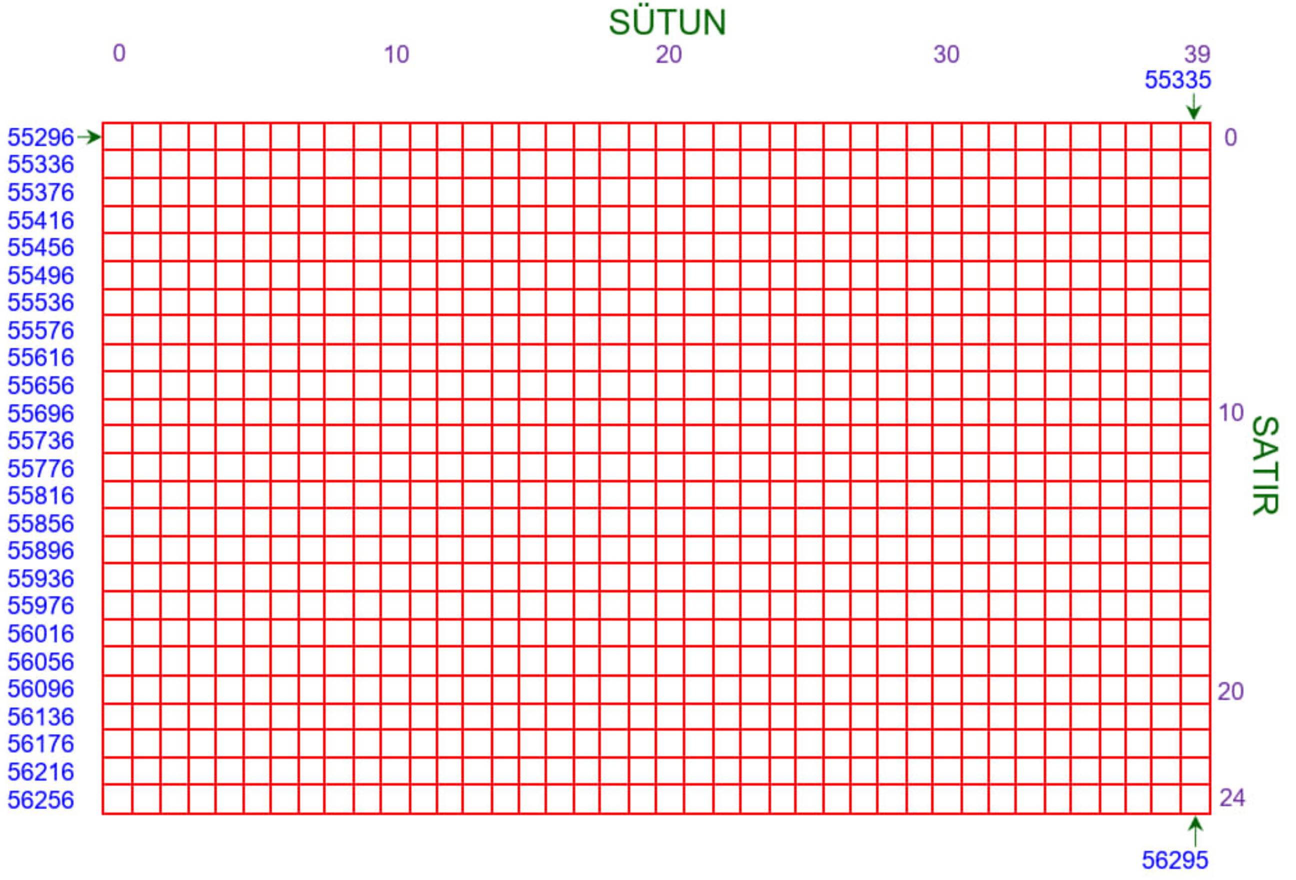


Bir karakterin rengini değiştirmek için, renk belleğindeki bir yerleşime POKE komutu kullanarak yerleştirmeniz gereken değerler:

0 SİYAH
1 BEYAZ
2 KIRMIZI
3 TURKUAZ
4 EFLATUN
5 YEŞİL
6 MAVİ
7 SARI
8 TURUNCU
9 KAHVERENGİ
10 Açık KIRMIZI
11 GRİ 1
12 GRİ 2
13 Açık YEŞİL
14 Açık MAVİ
15 GRİ 3

Örneğin; ekranın sol üst köşesinde yer alan karakterin rengini kırmızıya dönüştürmek için: POKE 55296,2 yazmanız gerekir.

RENK BELLEK HARİTASI



EK E
MÜZİK-NOTA DEĞERLERİ

Bu ek, notalara birer numara vererek, gerçek notaların ve istenilen notanın elde edilmesi için ses çipinin, YÜKSEK FREKANS ve ALÇAK FREKANS kayıtlarına POKE komutu ile yerleştirilecek olan değerlerin tam bir listesini içerir.

MÜZİK NOTASI		OSİLATÖR FREKANS (NTSC)			OSİLATÖR FREKANS (PAL)		
NOTA	OKTAV	ONLU	YÜKSEK	ALÇAK	ONLU	YÜKSEK	ALÇAK
0	C-0	268	1	12	278	1	22
1	C#-0	284	1	28	294	1	38
2	D-0	301	1	45	312	1	56
3	D#-0	318	1	62	331	1	75
4	E-0	337	1	81	350	1	94
5	F-0	358	1	102	371	1	115
6	F#-0	379	1	123	393	1	137
7	G-0	401	1	145	417	1	161
8	G#-0	425	1	169	441	1	185
9	A-0	451	1	195	468	1	212
10	A#-0	477	1	221	496	1	240
11	B-0	506	1	250	525	2	13
16	C-1	536	2	24	556	2	44
17	C#-1	568	2	56	589	2	77
18	D-1	602	2	90	625	2	113
19	D#-1	637	2	125	662	2	150
20	E-1	675	2	163	701	2	189
21	F-1	716	2	204	743	2	231
22	F#-1	758	2	246	787	3	19
23	G-1	803	3	35	834	3	66
24	G#-1	851	3	83	883	3	115
25	A-1	902	3	134	936	3	168
26	A#-1	955	3	187	992	3	224
27	B-1	1012	3	244	1051	4	27
32	C-2	1072	4	48	1113	4	89
33	C#-2	1136	4	112	1179	4	155
34	D-2	1204	4	180	1250	4	226
35	D#-2	1275	4	251	1324	5	44
36	E-2	1351	5	271	1403	5	123
37	F-2	1432	5	152	1486	5	206
38	F#-2	1517	5	237	1575	6	39

MÜZİK NOTASI		OSİLATÖR FREKANS (NTSC)			OSİLATÖR FREKANS (PAL)		
NOTA	OKTAV	ONLU	YÜKSEK	ALÇAK	ONLU	YÜKSEK	ALÇAK
39	G-2	1607	6	71	1668	6	132
40	G#-2	1703	6	167	1767	6	231
41	A-2	1804	7	12	1873	7	81
42	A#-2	1911	7	119	1984	7	192
43	B-2	2025	7	233	2102	8	54
48	C-3	2145	8	97	2227	8	179
49	C#-3	2273	8	225	2359	9	55
50	D-3	2408	9	104	2500	9	196
51	D#-3	2551	9	247	2649	10	89
52	E-3	2703	10	143	2806	10	246
53	F-3	2864	11	48	2973	11	157
54	F#-3	3034	11	218	3150	12	78
55	G-3	3215	12	143	3337	13	9
56	G#-3	3406	13	78	3535	13	207
57	A-3	3608	14	24	3746	14	162
58	A#-3	3823	14	239	3969	15	129
59	B-3	4050	15	210	4205	16	109
64	C-4	4291	16	195	4455	17	103
65	C#-4	4547	17	195	4719	18	111
66	D-4	4817	18	209	5000	19	136
67	D#-4	5103	19	239	5298	20	178
68	E-4	5407	21	31	5613	21	237
69	F-4	5728	22	96	5946	23	58
70	F#-4	6069	23	181	6300	24	156
71	G-4	6430	25	30	6675	26	19
72	G#-4	6812	26	156	7071	27	159
73	A-4	7217	28	49	7492	29	68
74	A#-4	7647	29	223	7938	31	2
75	B-4	8101	31	165	8410	32	218
80	C-5	8583	33	135	8910	34	206
81	C#-5	9094	35	134	9439	36	223
82	D-5	9634	37	162	10001	39	17
83	D#-5	10207	39	223	10596	41	100
84	E-5	10814	42	62	11226	43	218
85	F-5	11457	44	193	11893	46	117
86	F#-5	12139	47	107	12600	49	56

MÜZİK NOTASI		OSİLATÖR FREKANS (NTSC)			OSİLATÖR FREKANS (PAL)		
NOTA	OKTAV	ONLU	YÜKSEK	ALÇAK	ONLU	YÜKSEK	ALÇAK
87	G-5	12860	50	60	13350	52	38
88	G#-5	13625	53	57	14143	55	63
89	A-5	14435	56	99	14985	58	137
90	A#-5	15294	59	190	15876	62	4
91	B-5	16203	63	75	16820	65	180
96	C-6	17167	67	15	17820	69	156
97	C#-6	18188	71	12	18879	73	191
98	D-6	19269	75	69	20002	78	34
99	D#-6	20415	79	191	21192	82	200
100	E-6	21629	84	125	22452	87	180
101	F-6	22915	89	131	23787	92	235
102	F#-6	24278	94	214	25201	98	113
103	G-6	25721	100	121	26700	104	76
104	G#-6	27251	106	115	28287	110	127
105	A-6	28871	112	199	29970	117	18
106	A#-6	30588	119	124	31752	124	8
107	B-6	32407	126	151	33640	131	104
112	C-7	34334	134	30	35640	139	56
113	C#-7	36376	142	24	37759	147	127
114	D-7	38539	150	139	40005	156	69
115	D#-7	40830	159	126	42384	165	144
116	E-7	43258	168	250	44904	175	104
117	F-7	45830	179	6	47574	185	214
118	F#-7	48556	189	172	50403	196	227
119	G-7	51443	200	243	53400	208	152
120	G#-7	54502	212	230	56575	220	255
121	A-7	57743	225	143	59940	234	36
122	A#-7	61176	238	248	63504	248	16
123	B-7	64814	253	46	-	-	-

FİLTRE AYARLARI

KONUM	İÇERİĞİ
54293	Alçak kesim frekansı (0-7)
54294	Yüksek kesim frekansı (0-255)
53295	Salınım (bit 4-7)
	Ses 3 filtresi (2'inci bit)
	Ses 2 filtresi (1'inci bit)
	Ses 1 filtresi (0'ıncı bit)
54296	Yüksek frekans geçişi (6’ncı bit)
	Ara frekans geçişi (5'inci bit)
	Alçak frekans geçişi (4'üncü bit)
	Volüm (0-3'üncü bitler)

EK F
BİBLİYOGRAFİ

ISBN	BAŞLIK	YAZAR
Yayımcı: Addison-Wesley		
9780201015898	**BASIC and the Personal Computer**	Dwyer, Thomas A.; Critchfield, Margot
Yayımcı: Compute! Books		
9780942386011	**Compute!'s First Book Of PET/CBM**	Lock, Robert
9780942386042	**Programming the PET/CBM**	West, Raeto C.
Yayımcı: Cow Bay Computing		
	Feed Me, I'm Your PET Computer	Alexander, Carole
	Looking Good with Your PET	Alexander, Carole
	Teacher's PET – Plans, Quizzes, and Answers	
Yayımcı: Creative Computing		
9780916688288	**Getting Acquainted With Your VIC-20**	Hartnell, Tim
Yayımcı: Dilithium Press		
9780918398253	**32 BASIC Programs for the PET Computer**	Rugg, Tom; Feldman, Phil
Yayımcı: Hayden Books		
9780810455344	**BASIC Conversions Handbook for Apple TRS-80 and PET Users**	Brain, David A.; Oviate, Philip R.; Paquin, Paul J.A.; Stone Jr, Chandler D.
9780810457607	**BASIC From The Ground Up**	Simon, David E
9780810410503	**Library of PET Subroutines**	Hampshire, Nick
Yayımcı: Little, Brown		
9780876261477	**The Computer Tutor: Learning Activities For Homes and Schools**	Orwig, Gary W.; Hodges, W.
Yayımcı: McGraw-Hill Osborne Media		
9780931988752	**CBM Professional Computer Guide**	Osborne, Adam
9780070491571	**Hands-On BASIC with a PET**	Peckham, Herbert D.
9780931988820	**Osborne CP/M User Guide**	Hogan, Thom
9780931988318	**PET and the IEEE 488 Bus (GPIB)**	E. R Fisher; C W Jensen

ISBN	BAŞLIK	YAZAR
9780931988707	**PET Fun And Games**	Jeffries, Ron; Fisher, Glen
9780931988554	**PET/CBM Personal Computer Guide**	Osborne, Adam
9780931988400	**Some Common BASIC Programs: Commodore PET/CBM Edition**	Poole, Lon
9780931988295	**The 8086 Book**	Rector, Russell
9780931988509	**VisiCalc: Home and Office Companion**	Castlewitz, D.M.
Yayımcı: MOS Technology Inc.		
	MCS6500 Microcomputer Family Hardware Manual	MOS Technology Inc.
Yayımcı: Prentice Hall		
9780136617693	**PET/CBM BASIC**	Haskell, Richard E
9780136618355	**The PET Personal Computer for Beginners**	Dunn, Seamus
9780835983839	**VIC Games and Recreations**	Camora, Dorothy
Yayımcı: Reston Publishing		
9780835955256	**PET BASIC: Training Your PET Computer**	Zamora, Ramon
9780835955300	**PET Games and Recreations**	Oglesby, Mac.; Lindsay, Len.; Kunkin, Dorothy B.
Yayımcı: Sams Publishing		
9780672219856	**The Howard W. Sams Crash Course In Microcomputers**	Frenzel, Louis E.
9780810461864	**I Speak BASIC To My PET**	Jones, A
9780672217906	**Mostly BASIC: Applications for your PET**	Berenbon, Howard
9780810410510	**PET Graphics**	Hampshire, Nick
9780672217951	**PET Interfacing**	Downey, James M
9780672219481	**VIC 20 Programmer's Reference Guide**	Finkel, A; Higginbottom, P; Harris, N; Tomczyk, M
Yayımcı: Tab Books		
9780830615216	**Basic, BASIC-English Dictionary for the Apple, PET, and TRS-80**	Noonan, Larry
Yayımcı: Total Information Services		
	Understanding Your PET/CBM	
	Understanding Your VIC	Schultz, David
Yayımcı: Winthrop Publishers		
9780876261668	**Computer Games for Businesses, Schools, and Homes**	Nahigian, J. Victor

Commodore Dergileri Commodore 64'ünüz için size en güncel bilgileri sağlar. Abone olmayı ciddi olarak düşünmeniz gereken en popüler yayınlardan ikisi şunlardır:

COMMODORE – Microcomputer Magazine iki ayda bir yayınlanır ve abonelikle edinilebilir (ABD için yıllık 15,00 ABD Doları ve dünya çapında yıllık 25,00 ABD Doları).

POWER/PLAY – Ev Bilgisayarı Dergisi üç ayda bir yayınlanır ve abonelikle edinilebilir (ABD için yıllık 10,00 ABD Doları ve dünya çapında yıllık 15,00 ABD Doları).

EK G
VIC ÇİPİ KAYIT HARİTASI
53248 ($D000) Başlangıç (Taban) Adresi

Kayıt No		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Onlu	Onaltılı									
0	0	S0X7							S0X0	Yaratık 0'ın X konumu
1	1	S0Y7							S0Y0	Yaratık 0'ın Y konumu
2	2	S1X7							S1X0	Yaratık 1'in X konumu
3	3	S1Y7							S1Y0	Yaratık 1'in Y konumu
4	4	S2X7							S2X0	Yaratık 2'nin X konumu
5	5	S2Y7							S2Y0	Yaratık 2'nin Y konumu
6	6	S3X7							S3X0	Yaratık 3'ün X konumu
7	7	S3Y7							S3Y0	Yaratık 3'ün Y konumu
8	8	S4X7							S4X0	Yaratık 4'ün X konumu
9	9	S4Y7							S4Y0	Yaratık 4'ün Y konumu
10	A	S5X7							S5X0	Yaratık 5'in X konumu
11	B	S5Y7							S5Y0	Yaratık 5'in Y konumu
12	C	S6X7							S6X0	Yaratık 6'nın X konumu
13	D	S6Y7							S6Y0	Yaratık 6'nın Y konumu
14	E	S7X7							S7X0	Yaratık 7'nin X konumu
15	F	S7Y7							S7Y0	Yaratık 7'nin Y konumu
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	X koordinatın en soldaki biti
17	11	RC8	ECM	BMM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	Y kaydırma modu
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Izgara
19	13	LPX7							LPX0	Işık-Kalem X
20	14	LPY7							LPY0	Işık-Kalem Y
21	15	SE7							SE0	Yaratık aktif (Açık/Kapalı)
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	X kaydırma modu
23	17	SEXY7							SEXY0	Y Yaratık genişleme
24	18	VS13	VS12	VS11	VS10	CB13	CB12	CB11	N.C.	Ekran, karakter ve bellek
25	19	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRQ	Kesinti isteği
26	1A	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	Kesinti isteği maskeleme
27	1B	BSP7							BSP0	Zemin yaratık önceliği
28	1C	SCM7							SCM0	Çok-renkli yaratık seçimi
29	1D	SEXX7							SEXX0	X yaratık genişleme
30	1E	SSC7							SSC0	Yaratık-yaratık çarpışma
31	1F	SBC7							SBC0	Yaratık-zemin çarpışma

KAYIT NO		
ONLU	ONALTILI	RENK
32	20	Sınır rengi
33	21	Zemin rengi 0
34	22	Zemin rengi 1
35	23	Zemin rengi 2
36	24	Zemin rengi 3
37	25	Çok-renkli yaratık 0
38	26	Çok-renkli yaratık 1

KAYIT NO		
ONLU	ONALTILI	RENK
39	27	Yaratık rengi 0
40	28	Yaratık rengi 1
41	29	Yaratık rengi 2
42	2A	Yaratık rengi 3
43	2B	Yaratık rengi 4
44	2C	Yaratık rengi 5
45	2D	Yaratık rengi 6
46	2E	Yaratık rengi 7

RENK KODLARI

ONLU	ONALTILI	RENK
0	0	Siyah
1	1	Beyaz
2	2	Kırmızı
3	3	Camgöbeği
4	4	Mor
5	5	Yeşil
6	6	Mavi
7	7	Sarı

ONLU	ONALTILI	RENK
8	8	Turuncu
9	9	Kahverengi
10	A	Açık kırmızı
11	B	Gri 1
12	C	Gri 2
13	D	Açık yeşil
14	E	Açık mavi
15	F	Gri 3

EFSANE:
Çok renkli karakter modunda sadece 0 ila 7 arası renkler kullanılabilir.

EK H

MATEMATİKSEL FONKSİYONLAR

Fonksiyonların Commodore 64 BASIC'indeki kullanımları:

FONKSİYON	BASIC KARŞILIĞI
SEKANT	SEC(X)=1/COS(X)
KOSEKANT	CSC(X)=1/SIN(X)
KOTANJANT	COT(X)=1/TAN(X)
TERS SİNÜS	ARCSIN(X)=ATN(X/SQR(-X*X+1))
TERS KOSİNÜS	ARCCOS(X)=-ATN(X/SQR(-X*X+1))+π/2
TERS SEKANT	ARCSEC(X) = ATN(X/SQR(X*X-1))
TERS KOSEKANT	ARCSEC(X)= ATN(X/SQR(X*X-1))+(SGN(X)-1)*π/2
TERS KOTANJANT	ARCOT(X)=ATN(X)+π/2
HİPERBOLİK SİNÜS	SINH(X)=(EXP(X)-EXP(-X))/2
HİPERBOLİK KOSİNÜS	COSH(X)=(EXP(X)+EXP(-X))/2
HİPERBOLİK TANJANT	TANH(X)= EXP(-X)/(EXP(x)+EXP(-X))*2+ 1
HİPERBOLİK SEKANT	SECH(X)=2/(EXP(X)+EXP(-X))
HİPERBOLİK KOSEKANT	CSCH(X)=2/(EXP(X)-EXP(-X))
HİPERBOLİK KOTANJANT	COTH(X)=EXP(-X)/(EXP(X)-EXP(-X))*2+1
TERS HİPERBOLİK SİNÜS	ARCSINH(X)=LOG(X+SQR(X*X+1))
TERS HİPERBOLİK KOSİNÜS	ARCCOSH(X)=LOG(X+SQR(X*X-1))
TERS HİPERBOLİK TANJANT	ARCTANH(X)=LOG((1+X)/(1-X))/2
TERS HİPERBOLİK SEKANT	ARCSECH(X)= LOG(SQR(1-X*X+1)+1/X)
TERS HİPERBOLİK KOSEKANT	ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1/X)
TERS HİPERBOLİK KOTANJANT	ARCCOTH(X)= LOG((X+1)/(X-1))/2

EK I

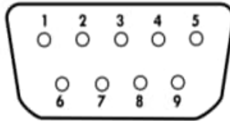
GİRİŞ/ÇIKIŞ CİHAZLARI İÇİN UÇ ÇIKIŞLARI

Bu bölüm Commodore 64'e yapılabilecek bağlantıları göstermek için düzenlenmiştir:

1) Oyun Giriş/Çıkış
2) Kartuş Girişi
3) Audio/Video
4) Seri Giriş/Çıkış (Disk/Yazıcı)
5) Modülatör Çıkışı
6) Teyp
7) Kullanıcı Portu

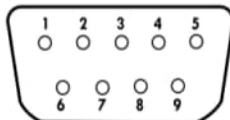
Kontrol Portu 1

BACAK	TİP	NOT
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY	
6	BUTTON A/LP	
7	+5V	MAX. 50mA
8	GND	
9	POT AX	



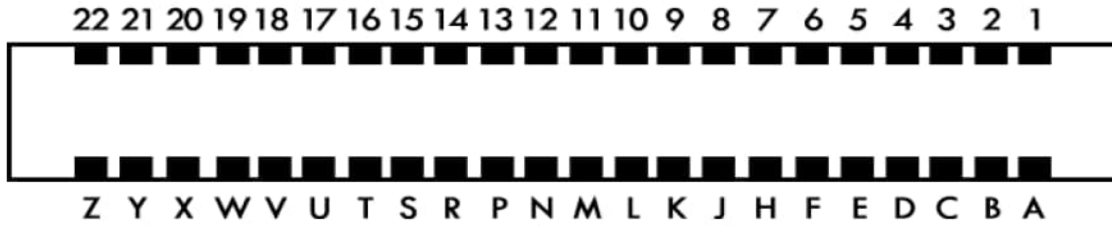
Kontrol Portu 2

BACAK	TİP	NOT
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY	
6	BUTTON B	
7	+5V	MAX. 50mA
8	GND	
9	POT BX	



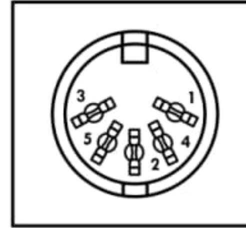
Kartuş Girişi

BACAK	TİP	BACAK	TİP	BACAK	TİP	BACAK	TİP
1	GND	12	BA	A	GND	N	A9
2	+5V	13	DMA	B	ROMH	P	A8
3	+5V	14	D7	C	RESET	R	A7
4	IRQ	15	D6	D	NMI	S	A6
5	R/W	16	D5	E	S02	T	A5
6	Dot Clock	17	D4	F	A15	U	A4
7	I/O 1	18	D3	H	A14	V	A3
8	GAME	19	D2	J	A13	W	A2
9	EXROM	20	D1	K	A12	X	A1
10	I/O 2	21	D0	L	A11	Y	A0
11	ROML	22	GND	M	A10	Z	GND



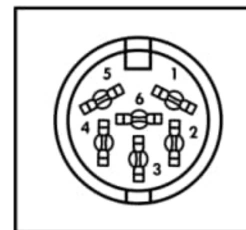
Audio/Video

BACAK	TİP
1	LUMINANCE
2	GND
3	AUDIO OUT
4	VIDEO OUT
5	AUDIO IN



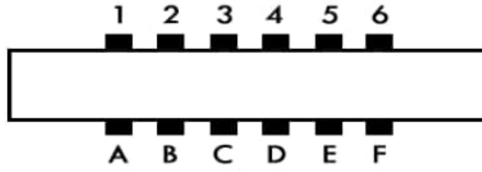
Seri Giriş/Çıkış

BACAK	TİP
1	SERIAL SRQIN
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	RESET



Teyp

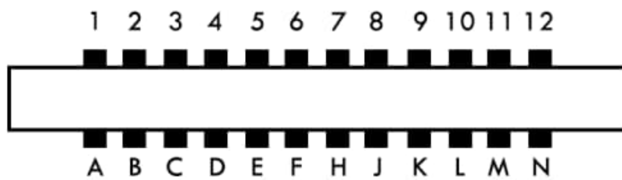
BACAK	TİP
A-1	GND
B-2	+5V
C-3	Cassette Motor
D-4	Cassette Read
E-5	Cassette Write
F-6	Cassette Sense



Kullanıcı Giriş/Çıkış

BACAK	TİP	NOT
1	GND	
2	+5V	MAX. 100mA
3	RESET	
4	CNTI	
5	SP1	
6	CNT2	
7	SP2	
8	PC2	
9	SER. ATN IN	
10	9 VAC	MAX. 100mA
11	9 VAC	MAX. 100mA
12	GND	

BACAK	TİP	NOT
A	GND	
B	FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	



EK J
STANDART BASIC PROGRAMLARININ COMMODORE 64 BASIC'İNE ÇEVRİLMESİ

Eğer Commodore BASIC'inden farklı bir BASIC ile yazılmış programlarınız varsa, birkaç ufak değişiklikle bu programları Commodore 64'te işleyebilir hale getirebilirsiniz. Bu değişiklikleri yapmanıza yardımcı olacak birkaç ipucunu bu ekte bulabilirsiniz.

Yazısal Dizi Boyutları

Yazısal dizilerin boyutlarını tanımlamakta kullanılan tüm yönergeleri çıkarın. I uzunluğunda J elemanlı bir yazısal dizinin tanımlanmasında kullanılan DIM A$(I,J) yönergesi Commodore BASIC'ine DIM A$(J) olarak çevrilir.

Bazı BASIC'ler yazısal dizilerin birleştirilmesinde (concatenation) virgül ya da kesme işareti kullanırlar. Commodore 64 BASIC'inde bu işlem artı işareti ile yapılır. Bu yüzden bu tur işaretleri artı olacak şekilde değiştirin.

Commodore 64 BASIC'inde yazısal dizilerden küçük yazısal dizi setleri oluşturmak için MID$, RIGHT$, LEFT$ gibi fonksiyonlar kullanılır. A$(I)'yı, A$'ın I'ncı karakterine erişebilecek şekilde düzenleyin ya da A$'in I pozisyonundan J pozisyonuna kadar olan bir küçük yazısal dizi oluşturmak için aşağıdaki biçimleri deneyin:

Diğer BASIC'te	Commodore 64 BASIC'inde
A$(I)=X$	A$ = LEFT$(A$,I-1) + X$ + MID$(A$,I+1)
A$(I,J)=X$	A$ = LEFT$(A$,I-1) + X$ + MID$(A$,J+1)

Çoklu Değer Atamaları

B ve C'yi 0'a eşitlemek için bazı BASIC'lerde aşağıdaki gibi yazılan yönergelere izin verilir:

```
10 LET B = C = 0
```

Commodore 64 BASIC'i ikinci eşit işaretini mantıksal operatör olarak algılayacak ve eğer C = 0 ise B = -1 olacaktır. Bunun için yukarıda verilen yönergenin aşağıdaki gibi değiştirilmesi gerekir:

```
10 C = 0:B = 0
```

Çoklu Yönergeler

Bazı BASIC'lerde birden fazla yönergeyi birbirinden ayırmak için"\" işareti kullanılır. Commodore 64 BASIC'inde yönergeleri birbirinden ayırmak için ":" kullanılır.

MAT Fonksiyonu

Bazı BASIC'ler için geçerli olan MAT fonksiyonları Commodore BASIC'i için FOR ... NEXT döngülerine dönüştürülmelidirler.

EK K

HATA UYARILARI

Bu ek, Commodore 64 tarafından verilen hata uyarılarının tam bir listesini verir. Listede uyarıların oluşum sebepleri de gösterilmiştir.

BAD DATA (Yanlış veri): Program sayısal veri (DATA) beklerken, açık olan dosyadan yazısal dizi verisi alınmıştır.

BAD SUBSCRIPT (Yanlış boyut): Program DIM yönergesiyle belirlenmiş olan sınırın üzerinde bir dizi elemanına ulaşmaya çalışmıştır.

BREAK (Kesilme): Programın çalışması **RUN/STOP** tuşuna bastığınız için durdurulmuştur.

CAN'T CONTINUE (Devam edemez): CONT komutu, ya program daha önce hiç RUN edilmediği, hata uyarısıyla kesildiği, ya da bir satırı değiştirildiği için devam edemez.

DEVICE NOT PRESENT (Cihaz işler durumda değil): OPEN, CLOSE, CMD, PRINT#, INPUT# veya GET# için gerekli olan yan birim (örneğin disk sürücü veya yazıcı) açık değil.

DIVISION BY ZERO (Sıfırla bölme): 0'a bölünme matematiksel olarak yanlıştır ve buna izin verilmez.

EXTRA IGNORED: INPUT yönergesine karşılık olarak çok fazla veri öğesi girildiğinde oluşur. Sadece INPUT yönergesini izleyen değişken sayısı kadar öğe değerlendirilir.

FILE NOT FOUND (Dosya bulunamadı): Teyp üzerindeki bir dosyayı ararken. END-OF-TAPE (Bantın sonu) işareti bulunduysa, disk üzerindeki bir dosyayı ararken de verilen isimde bir dosya bulunmamışsa bu hata oluşur.

FILE NOT OPEN (Dosya açık değil): CLOSE, CMD, PRINT#, INPUT# veya GET# komutlarında belirtilen dosyaların, daha önceden OPEN komutuyla açılmış olması gerekir.

FILE OPEN (Dosya açık): Daha önce açılmış olan bir dosya ile aynı numarada yeni bir dosya açma girişiminde bulunulmuştur.

FORMULA TOO COMPLEX (Formül çok karışık): Değerlendirilmesi gereken formül en az ikiye ayrılmalı, ya da formülde çok fazla parantez var.

ILLEGAL DIRECT (Geçersiz komut): INPUT ve GET yönergeleri sadece program içinde kullanılabilir, doğrudan modda kullanılamaz.

ILLEGAL OUANTITY (Geçersiz nicelik): Formül içinde veya bir yönergenin parametresi olarak kullanılan değer, geçerli alt ve üst limitlerin dışındadır.

LOAD (Yükleme hatası): Kaset veya disket üzerindeki programın yüklenmesi sırasında bir sorun var.

NEXT WITHOUT FOR (FOR'suz NEXT hatası): Yanlış olarak iç içe girilmiş döngüler var veya yönergede kullanılan değişken ismine karşılık gelen FOR yönergesi yok.

NOT INPUT FILE (Girdi dosyası değil): Yalnızca çıktı dosyası olarak açılmış bir dosyaya, INPUT veya GET komutlarını kullanarak veri girişi yapmaya çalışıldığında.

NOT OUTPUT FILE (Çıktı dosyası değil): Yalnızca girdi dosyası olarak tanımlanmış olan dosyaya PRINT komutu ile veri yazdırma girişiminde.

OUT OF DATA (DATA kalmadı): READ (oku) yönergesi ile okunması gereken veriler tükenmiştir.

OUT OF MEMORY (Yetersiz bellek): Program veya değişkenler için yeterli RAM bellek kalmamıştır veya programda iç içe çok fazla FOR … NEXT veya GOSUB … RETURN döngüsü var.

OVERFLOW (Taşma): Yapılmakta olan işlemin sonucu, izin verilen en büyük sayı olan 1.70141884E + 38'den büyük.

REDIM-D ARRAY (Dizi yeniden boyutlandı): Bir dizi sadece bir kere boyutlanır (DIM yönergesi ile). Eğer bir dizi değişkeni boyutlanmadan kullanıldıysa, otomatik bir DIM işlemi, dizi elemanlarının sayısını 10 olarak belirler ve sonradan gelen herhangi bir DIM yönergesi hataya neden olur.

REDO FROM START (Baştan yap): INPUT yönergesinde sayısal veri beklenirken, harf girişi yapılırsa. Doğru olan girişi yapınca, program kendiliğinden devam eder.

RETURN WITHOUT GOSUB (GOSUB'sız RETURN): Program, GOSUB ile gelmediği bir noktada RETURN yönergesiyle karşılaşmıştır.

STRING TOO LONG (Dizi çok uzun): Bir yazısal dizi en fazla 255 karakter içerebilir.

?SYNTAX ERROR (Yazım hatası): Verilen bir yönerge Commodore 64 tarafından tanınmıyor. Unutulmuş veya fazladan konulmuş bir parantez, ya da bir komutun yanlış yazılması gibi.

TYPE MISMATCH (Tür uyuşmazlığı): Bu hata yanlış değişken türü (örneğin yazısal dizi yerine bir sayı) kullanıldığında ortaya çıkar.

UNDEF'D FUNCTION (Tanımlanmamış fonksiyon): Kullanıcı tanımlı bir fonksiyon kullanılmak istenmiş, fakat önceden DEF FN yönergesi ile tanımlanmamış.

UNDEF'D STATEMENT (Tanımlanmamış yönerge): Var olmayan bir satır numarasını GOTO, GOSUB veya RUN yönergelerinde kullanma girişiminde.

VERIFY (Doğrulama hatası): Bellekte o an var olan programla, disket veya kaset üzerindeki programın uyuşmaması sonucunda ortaya çıkar. Program disket veya kasete yanlış kaydedilmiştir.

EK L

6510 MİKROİŞLEMCİNİN ÖZELLİKLERİ

TANIMLAMA

6510, düşük maliyetli mikrobilgisayar sistemidir. Bu sistem, küçük sistemlerin ve ek donanımların kontrolleri sırasında karşılaşılan geniş çaplı problemlerin çözümlerini getirebilecek yetenektedir.

Çipin üzerinde, 8 bitlik çift yönlü Giriş/Çıkış portu ve Veri-Yön Kaydı bulunur. Giriş/Çıkış portu, 0000 adresindeki Çıkış Kaydı üzerinde. Veri-Yön Kaydı ise 0001 adresinde yer almaktadır.

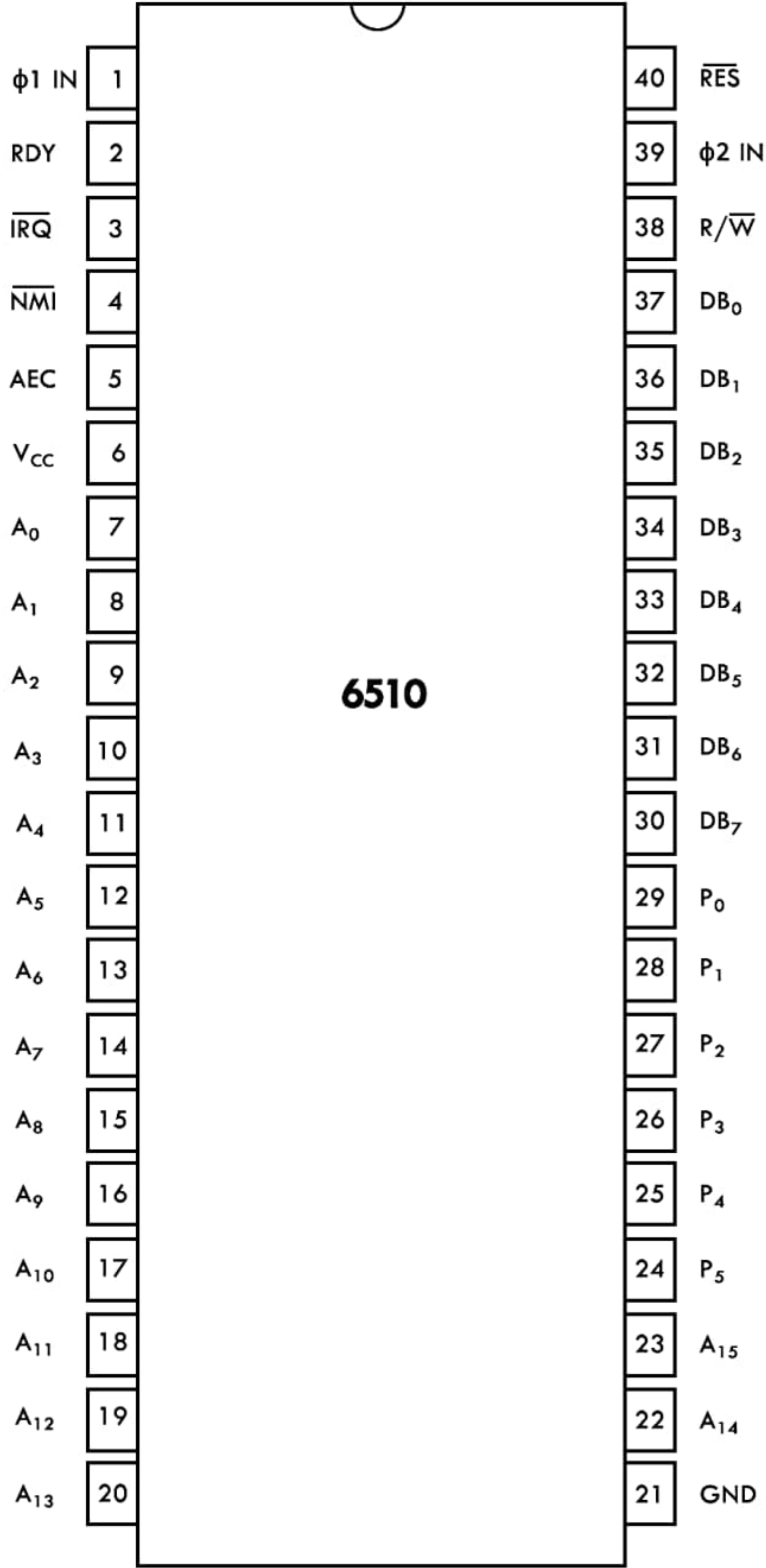
Üç-Durumlu 16 bitlik Adres Veri Yolu. Doğrudan Bellek Erişimine ve çok işlemcili sistemlerin belleği paylaşmalarına olanak sağlar.

İşlemcinin iç yapısı, yazılımların birbirlerine uyarlanabilmesi için MOS Teknolojisindeki 6502'ye benzer şekilde oluşturulmuştur.

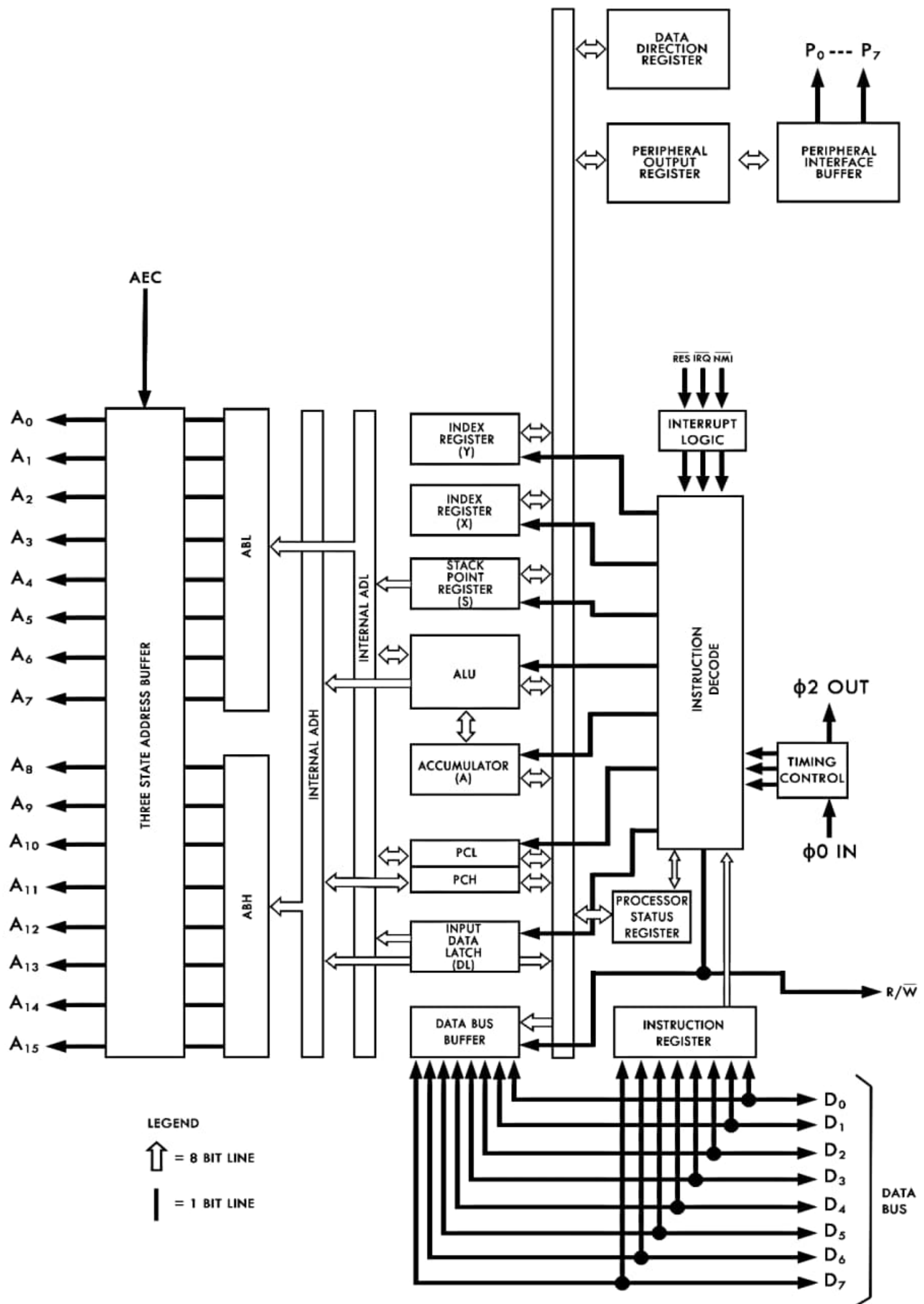
6510'UN ÖZELLİKLERİ

- Sekiz-Bitlik-Çift/Yönlü Giriş/Çıkış Portu
- +5 Voltluk tek kaynak
- N-kanal, silikon geçit, depletion load teknolojisi
- Sekiz-bit paralel işleme
- 56 Komut
- Onlu ve ikili aritmetik
- Onüç adresleme modu
- Doğru indeksleme yetisi
- Programlanabilir Yığın (Deste) göstergeci
- Değişken Yığın uzunluğu
- Kesinti yetisi
- Sekiz Bitlik Çift-Yönlü Veri Yolu
- 65K bayta kadar adreslenebilir bellek aralığı
- Doğrudan Bellek Erişimi yetisi
- M6800'e Veri Yolu uyarlanabilirliği
- Pipeline yapısı
- 1 MHz ve 2 MHz'de çalışabilme
- Herhangi bir tipte ya da hızda bellek kullanımı

BACAK DİZİLİMİ



6510 BLOK DİYAGRAM



6510 KARAKTERİSTİKLERİ

MAKSİMUM ORANLAR

ORAN	SEMBOL	DEĞER	BİRİM
VOLTAJ KAYNAĞI	V_{CC}	-0.3'den +7.0'a	V_{DC}
GİRİŞ VOLTAJI	V_{IN}	-0.3'den +7.0'a	V_{DC}
ÇALIŞMA ISISI	T_A	0'dan +70'e	°C
DEPOLAMA ISISI	T_{STG}	-55'den +150'e	°C

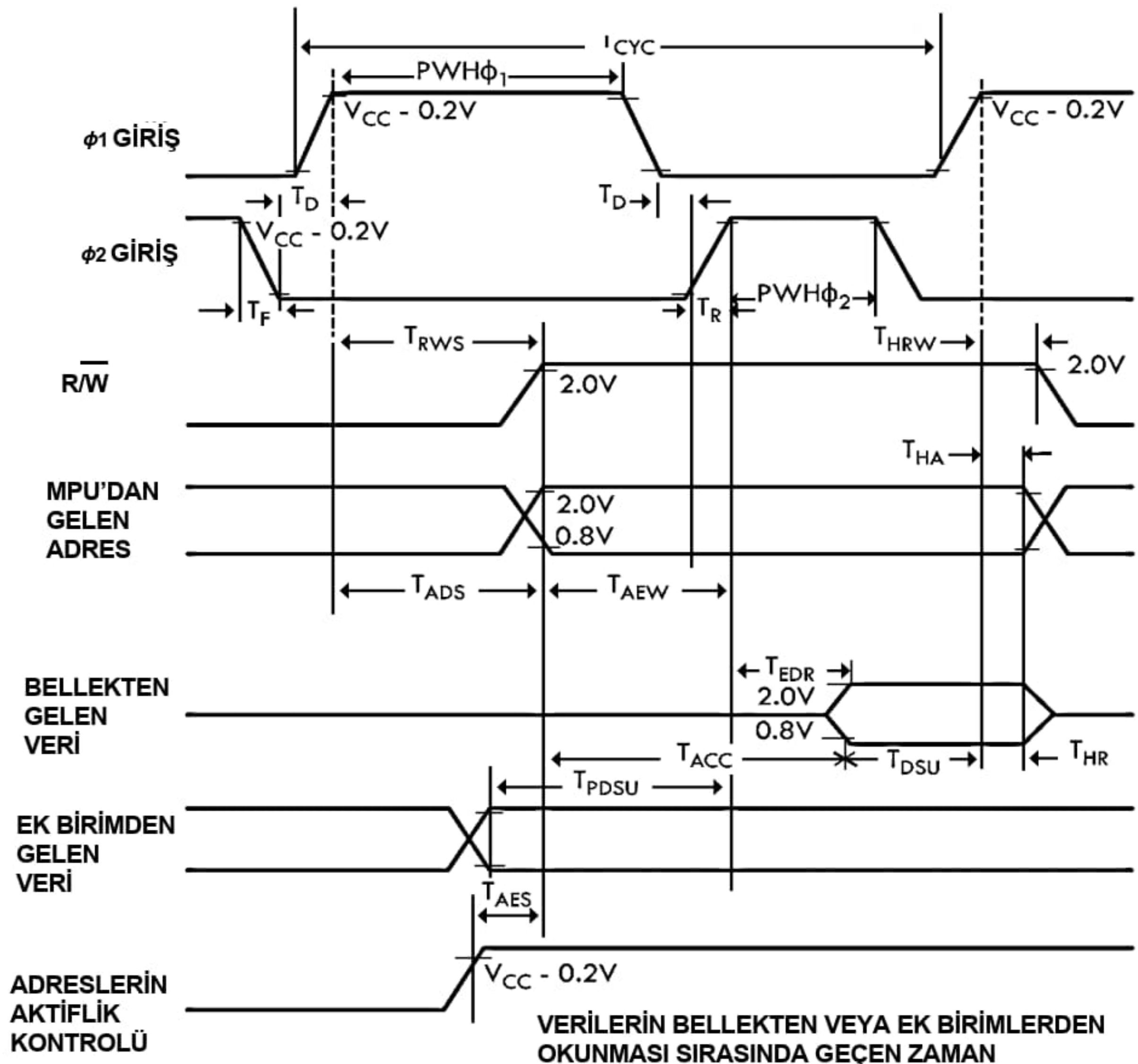
NOT: Bu cihaz, yüksek statik voltaj veya elektronik alanının oluşturabileceği zararlara karşılık bir iç koruma birimine sahiptir. Fakat gene de, güvenlik için yüksek voltaj uygulamasından kaçınılmalıdır.

ELEKTRİKSEL KARAKTERİSTİKLER
(V_{CC} = 5.0 V ±5%, V_{SS} = 0, T_A = 0°'den +70°C'ye)

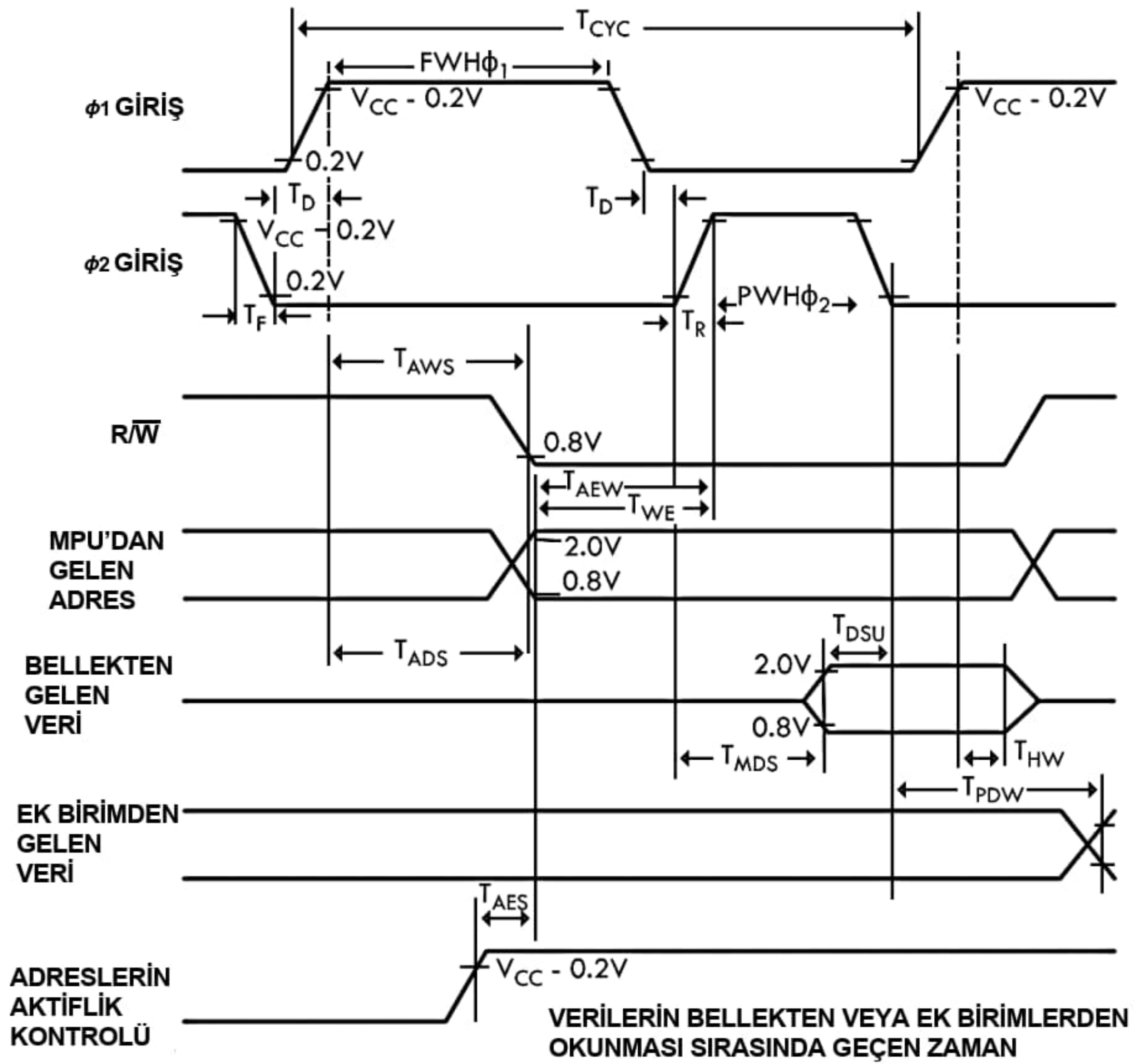
KARAKTERİSTİK	SEMBOL	MİNİMUM	TİP	MAKSİMUM	BİRİM
Yüksek Voltaj Girişi $\phi1$, $\phi2$(in) Yüksek Voltaj Girişi $\overline{RES}$, P_0-P_7, $\overline{IRQ}$, Veri	V_{IH}	V_{CC} - 0.2 V_{SS} + 2.0	— —	V_{CC} + 1.0V —	V_{DC} V_{DC}
Düşük Voltaj Girişi $\phi1$, $\phi2$(in) $\overline{RES}$, P_0-P_7, $\overline{IRQ}$, Veri	V_{IL}	V_{SS} - 0.3 —	— —	V_{SS} + 0.2 V_{SS} + 0.8	V_{DC} V_{DC}
Kaçak Akım Girişi (V_{IN} = 0'dan 5.25V'ye, V_{CC} = 5.25V) Mantık $\phi1$, $\phi2$(in)	I_{IN}	— —	— —	2.5 100	μA μA
Üç Durumlu (Kapalı Durum) Giriş Akımı (V_{IN} = 0.4'dan 2.4V'ye, V_{CC} = 5.25V) Veri Hatları	I_{TSI}	—	—	10	μA
Yüksek Voltaj Çıkışı (I_{OH} = -100μA_{DC}, VCC = 4.75V) Veri, A_0-A_{15}, O/Y, P_0-P_7	V_{OH}	V_{SS} + 2.4	—	—	V_{DC}

KARAKTERİSTİK	SEMBOL	MİNİMUM	TİP	MAKSİMUM	BİRİM
Düşük Voltaj Çıkışı (I_{OL} = 1.6mA_{DC}, V_{CC} = 4.75V) Veri, A_0-A_{15}, O/Y, P_0-P_7	V_{OL}	—	—	V_{SS} + 0.4	V_{DC}
Güç Kaynağı Akımı	I_{CC}	—	125		mA
Kapasitans (V_{IN} = 0, T_A = 25°C, f = 1MHZ)	C				pF
Mantık, P_0-P_7	C_{IN}	—	—	10	
Veri		—	—	15	
A_0-A_{15}, O/Y	C_{OUT}	—	—	12	
•1	C •1	—	30	50	
•2	C •2	—	50	80	

SAAT ZAMANLAMASI



SAAT ZAMANLAMASI



AC KARAKTERİSTİKLER

ELEKTRİKSEL KARAKTERİSTİKLER (V_{cc}=5V ±%5, V_{ss}=0 V, T_A=0°'dan 70°C'ye)

SAAT ZAMANLAMASI 1MHZ ZAMANLAMA 2MHZ ZAMANLAMA

KARAKTERİSTİK	SEMBOL	MİN.	TİP	MAK.	MİN.	TİP	MAK.	BİRİM
Döngü Süresi	T_{CYC}	1000	—	—	500	—	—	ns
Saat Darbe Genişliği ϕ1 (V_{CC} – 0,2V'de ölçüldü) ϕ2	PWH ϕ1 PWH ϕ2	430 470	— —	— —	215 235	— —	— —	ns ns
Düşüş Zamanı, Yükseliş Zamanı (0,2V'dan V_{CC} - 0,2V'a kadar ölçüldü)	T_F, T_R	—	—	25	—	—	15	ns
Saatler Arası Gecikme Süresi (0,2V'da Ölçülmüştür)	T_D	0	—	—	0	—	—	ns

KARAKTERİSTİK	SEMBOL	MİN.	TİP	MAK.	MİN.	TİP	MAK.	BİRİM
6508'den Okuma/Yazma Kurulum Süresi	T_{RWS}	—	100	300	—	100	150	ns
6508'den Adres Kurulum Süresi	T_{ADS}	—	100	300	—	100	150	ns
Bellek Okuma Erişim Süresi	T_{ACC}	—	—	575	—	—	300	ns
Veri İstikrar Süresi Dönemi	T_{DSU}	100	—	—	50	—	—	ns
Veri Tutma Süresi-Okuma	T_{HR}		—	—	—	—	—	ns
Veri Tutma Süresi-Yazma	T_{HW}	10	30	—	10	30	—	ns
6510'dan Veri Kurulum Süresi	T_{MDS}	—	150	200	—	75	100	ns
Adres Tutma Süresi	T_{HA}	10	30		10	30	—	ns
R/W Tutma Süresi	T_{HRW}	10	30	—	10	30	—	ns
Gecikme Süresi, Adres ϕ2 Pozitif Geçişinde Geçerlidir.	T_{AEW}	180	—	—	—	—	—	ns
Gecikme Süresi, ϕ2 Pozitif Geçişi Veri Geçerli.	T_{EDR}	—	—	395	—	—	—	ns
Gecikme Süresi, Veri ϕ2 Negatif Geçişine Kadar Geçerlidir	T_{DSU}	300	—	—	—	—	—	ns
Gecikme Süresi, R/W Negatif Geçişten ϕ2 Pozitif Geçişe	T_{WE}	130	—	—	—	—	—	ns
Gecikme Süresi, ϕ2 Çevresel Veri Negatif Geçiş Geçerli	T_{PDW}	—	—	1	—	—	—	µs
Çevresel Veri Kurulum Süresi	T_{PDSU}	300	—	—	—	—	—	ns
Adres Etkinleştirme Kurulum Süresi	T_{AES}	—	—	60	—	—	60	ns

SİNYALLERİN AÇIKLAMALARI

Saatler ($\phi 1$, $\phi 2$)

6510 için, V_{CC} voltaj düzeyinde çalışan iki-fazlı (çakışmayacak) saat gerekir.

Adres Veri Yolu (A_0-A_{15})

Bu çıkışlar, TTL'e uyarlanabilen, standart TTL yükleme ve 130 pf'ı sürebilecek yetidedir.

Veri Yolu (D_0-D_7)

Veri yolu için 8 uç kullanılır. Bu uçlar, cihazdan dışarıya ya da dışarıdan cihaza veri aktarımı sağlayabilecek şekilde çift-yönlü bir veri yoludur. Çıkışlar, standart bir TTL yükü ile 130 pf sürebilen üç-durumlu tamponlardır.

Reset

Bu giriş, reset için ya da mikroişlemcinin çalışmaya başlaması için kullanılır. Bu hattın alçak olarak tutulduğu zaman süresince, mikroişlemciye bir şey yazmak ya da mikroişlemcinin bir şey yazması yasaklanmıştır. Giriş üzerinde bir pozitif kenar algılandığında, mikroişlemci reset işlemlerine hemen başlayacaktır.

Sistemin başlatma süresi olan 6 saat çevrimden sonra, kesinti maskeleme bayrağı set edilir ve mikroişlemci program sayacına, bellek vektör yerleşimleri: FFFC ve FFFD yerleşimlerindeki değerleri yükleyecektir. Bu program kontrolü için başlangıç adresidir.

Power-up (güç sağlama) rutininde V_{CC} 4.75 volta eriştiğinde, reset, en az iki saat çevrimlik bir süre için alçak tutulmak zorundadır. Bu durumda, O/Y sinyali geçerli olacaktır.

İki saatlik çevrimden sonra reset, yüksek olduğunda mikroişlemci, yukarıda ayrıntılı olarak açıklanan normal reset işlemine devam edecektir.

Kesinti İsteği ($\overline{IRQ}$)

Bu TTL seviyeli giriş, mikroişlemci içinde bir kesinti dizisinin başlamasını istediğini belirtir. Mikroişlemci bu isteği değerlendirmeden önce, o anda işlemekte olduğu komutla işini bitirecektir. Bu durumda, Durum Kod Kaydındaki kesinti maskeleme biti incelenir. Eğer, kesinti maskeleme bayrağı ayarlanmamış ise, mikroişlemcide bir kesinti dizisi işleme sokulur. Program Sayacı ve işlemci Durum Kaydı, yığına aktarılır. Mikroişlemci kesinti maskeleme bayrağını yüksek olarak ayarlanır ve böylelikle daha başka kesintinin oluşmasına izin verilmez. Bu çevrimin sonunda, program sayacının alt baytına FFFE'deki değer, program sayacının üst baytına ise FFFF'deki değer yüklenir. Böylece program kontrolü, bu adreslerdeki bellek vektörüne aktarılır.

Adres Etkinleştirme Kontrolü (AEC)

Adres veri yolu, yalnızca Adres Etkinleştirme Kontrol hattı yüksek durumdayken geçerlidir. Alçak iken, Adres Veri Yolu yüksek-empedans durumundadır. Bu özellik, Doğrudan Bellek Erişimi ve çok-işlemcili sistemler için kolaylık sağlar.

Giriş/Çıkış Portu (P0-P7)

Çevre-birimleri portu için altı uç kullanılır ve bu port ile çevre birimlerine ya da çevrebirimlerinden veri transferi yapılır. Çıkış Kaydı, RAM'de 0001 adresinde. Veri Yön Kaydı da 0000 adresindedir. Çıkışlar bir standart TTL yükü ile 130 pf'ı sürebilecek özelliktedir.

Okuma/Yazma (R/W)

Bu sinyal, Veri Yolu üzerindeki verinin yönünü kontrol etmek amacıyla mikroişlemci tarafından üretilir. Mikroişlemcinin belleğe ya da bir çevre-birimi cihazına bilgi yazmasının dışında bu hat yüksek durumdadır.

ADRESLEME MODLARI

AKÜMÜLATÖR ADRESLEME

Bir baytlık komutlardır ve akümülatör ile yapılacak olan işlemi içerirler.

DOLAYSIZ ADRESLEME

Dolaysız adresleme, işlenen komutun ikinci baytıdır ve belleğin adreslenmesini gerektirmez.

MUTLAK ADRESLEME

Mutlak adreslemede komutun ikinci baytı, kullanılacak adresin sekiz en alt bitini (alt bayt) gösterir. Üçüncü bayt ise, üst sekiz bittir (üst bayt). Bu yüzden, mutlak adresleme modu, 65K baytlık adreslenebilir belleğin tümüne erişilmesini sağlar.

SIFIRINCI SAYFA ADRESLEME

Sıfırıncı sayfa komutlarında, daha kısa kodların kullanılması mümkündür ve komutun üst adres baytının sıfır kabul edilerek yalnızca ikinci baytının kullanılması yüzünden işletim sürelerde oldukça kısadır. Sıfırıncı sayfanın dikkatli kullanımı, kod verimliliği açısından oldukça faydalıdır ve önemli bir artış sağlar.

SIFIRINCI SAYFA, X VEYA Y İNDEKSLİ ADRESLEME

Bu modda sıfırıncı sayfa, indeks kaydı ile beraber kullanılır ve “sıfırıncı sayfa, X” ya da “sıfırıncı sayfa, Y” olarak belirtilir. Kullanılacak olan adres: ikinci baytın, indeks kaydında yer alan değerle toplanması ile hesaplanır. "Sıfırıncı sayfa” adresleme modunda, ilk bayt sıfır kabul edildiğinden, ikinci baytta yer alan değer sıfırıncı sayfadaki bir yerleşimin adresidir. Ek olarak, “sıfırıncı sayfa” adreslemesinin bu moddaki yapısına bağlı olarak, belleğin 8 üst bitine, elde (carry) değeri eklenmez ve sayfa sınırının aşılması diye bir sorunla karşılaşılmaz.

MUTLAK, X VEYA Y İNDEKSLİ ADRESLEME

Bu modda X ve Y indeksli mutlak adresleme, indeks kayıtları ile beraber kullanılır ve “mutlak, X” ve "mutlak, Y" olarak tanımlanır. Kullanılacak olan adres, X ve Y’nin içeriğinin, komutun ikinci ve üçüncü baytlarında belirtilen adrese eklenmesi ile bulunur. Bu mod, indeks kaydın indeks ya da bir sayma değerini, komutun da taban adresini içermesini sağlar. Bu tip bir indeksleme, herhangi bir yerleşimin ve indeksinin kullanılmasını ve böylece, kodlama ve işletim zamanının azaltılmasını sağlar.

İMALI ADRESLEME

Bu modda, komutlar her zaman aynı işi görürler. Adresin değişmesiyle işlevleri değişmez. İşlenen ve işlem, bu tek baytlık komutun içinde yer alır.

RELATİF ADRESLEME

Bu adresleme yalnızca dallanma komutları ile birlikte kullanılır ve koşullara bağlı bir dallanmada istenilen işlemi yerine getirir.

Komutun ikinci baytı. "Ofset” dediğimiz işlenendir. "Ofset", program sayacının bir sonraki komutu gösterdiği zaman içerdiği değerin en alt sekiz bitine eklenir.

Ofsetin aralığı, bir sonraki komutun -128 bayt öncesi ile komutun +127 bayt sonrasıdır.

İNDEKSLİ DOLAYLI ADRESLEME

(Dolaylı, X) diye tanımlanan bu adreslemede, komutun ikinci baytı. X indeks kaydındaki değere eklenir (elde biti hesaba katılmaz). Toplamın sonucu sıfırıncı sayfada yer alan bir bellek yerleşiminin adresidir. Ve burada yer alan değer kullanılacak olan adresin sekiz alt biti yani alt baytıdır.

Sıfırıncı sayfadaki bu yerleşimden sonra gelen yerleşimde ise kullanılacak olan adresin sekiz üst biti (üst baytı) yer alır. Kullanılacak adresin alt ve üst baytlarını içeren her iki bellek yerleşimi de sıfırıncı sayfada yer almak zorundadır.

DOLAYLI İNDEKSLİ ADRESLEME

(Dolaylı), Y diye belirtilen bu adreslemede komutun ikinci baytı, sıfırıncı sayfadaki bir yerleşimin adresidir. Bu yerleşimde yer alan değer, Y indeks kaydındaki değere eklenir. Sonuç, kullanılacak olan adresin en alt sekiz biti yani alt baytıdır. Bu toplam sonucundaki elde (carry) biti, sıfırıncı sayfada bir sonraki yerleşimde yer alan değere eklenir. Sonuç kullanılacak adresin sekiz üst biti yani üst baytıdır.

MUTLAK DOLAYLI

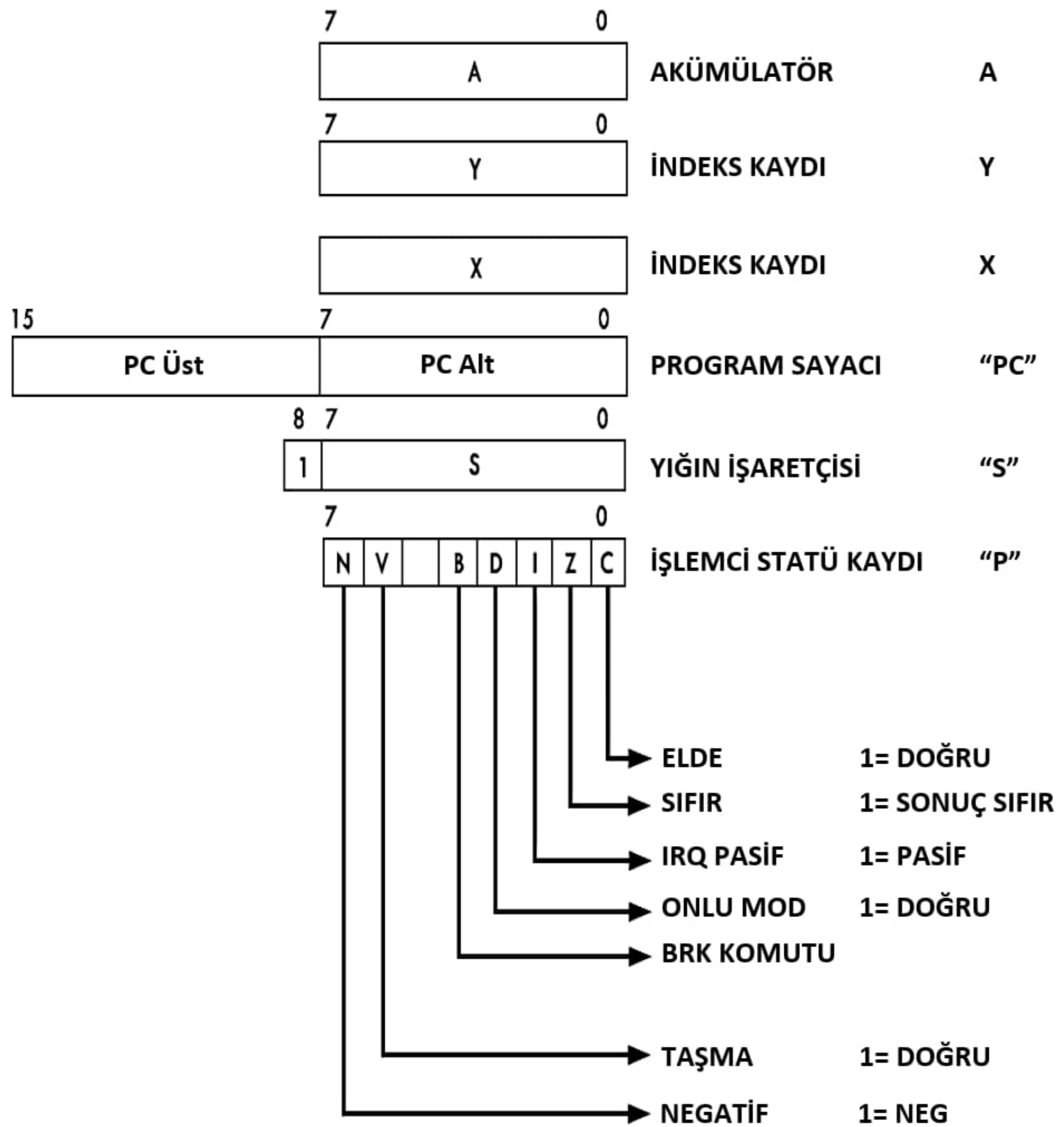
Komutun ikinci baytı, bellek yerleşiminin sekiz alt bitini içerir. Bu yerleşimin sekiz üst biti ise komutun üçüncü baytında yer alır. Tüm adresi böylece belirlenmiş olan bellek yerleşiminin içeriği, kullanılacak olan adresin sekiz alt bitidir. Bir sonraki bellek yerleşiminde ise kullanılacak olan adresin sekiz biti bulunur. Bu, program sayacının onaltı bitine aktarılır.

TALİMAT SETİ - ALFABETİK SIRALAMA

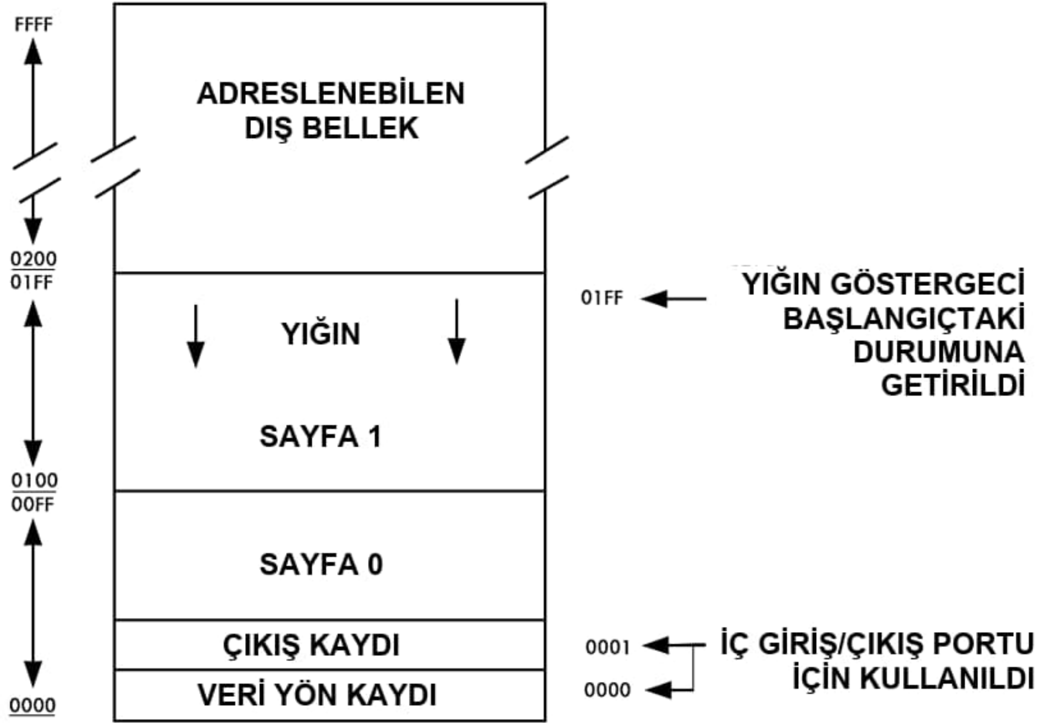
ADC	Belleği eldeki ile beraber akümülatöre ekle
AND	Belleği akümülatör ile "AND" (VE) işlemine sok
ASL	Bir bit sola kaydır (bellek veya akümülatörü)
BBC	Kalan sıfır ise dallan
BCS	Kalan sıfır değilse dallan
BEQ	Sonuç sıfır ise dallan
BIT	Bellekteki bitleri akümülatördekilerle karşılaştır.
BMI	Sonuç negatif ise dallan
BNE	Sonuç sıfır değil ise dallan
BPL	Sonuç pozitif ise dallan
BRK	Bitir
BVC	Taşma biti sıfır ise dallan
BVS	Taşma biti bir ise dallan
CLC	Elde bayrağını 0'la
CLD	Ondalık moddan çık
CLI	Kesinti önleme bitini sıfırla
CLV	Taşma bayrağını sıfırla
CMP	Bellekle akümülatörü karşılaştır
CPX	Bellekle X indeksini karşılaştır
CPY	Bellekle X indeksini karşılaştır
DEC	Bellekteki değeri bir azalt
DEX	X indeksini bir azalt
DEY	Y indeksini bir azalt

EOR Belleği akümülatörle "Exclusive-or” işlemine sok
INC Bellekteki değeri bir arttır
INX X indeksini bir arttır
INY Y indeksini bir arttır
JMP Belirtilen adrese atla
JSR Dönüş adresini saklayarak belirtilen adrese atla
LDA Bellekteki değeri akümülatöre yükle
LDX Bellekteki değeri X indeksine yükle
LDY Bellekteki değeri Y indeksine yükle
LSR Bir bit sağa kaydır (bellek veya akümülatörü)
NOP Hiçbir işlem yapma
ORA Belleği akümülatör ile "OR" (VEYA) işlemine sok
PHA Akümülatörü yığına gönder
PHP İşlemcinin statü değerini yığına gönder
PLA Akümülatörü yığından al
PLP İşlemcinin statü değerini yığından al
ROL Bir bit sola döndür (bellek veya akümülatörü)
ROR Bir bit sağa döndür (bellek veya akümülatörü)
RTI Kesinti işleminden geri dön
RTS Altprogramdan dön
SBC Ödünç (Borrow) ile birlikte belleği akümülatörden çıkart
SEC Kalan bayrağını 1 yap
SED Ondalık moduna gir
SEI Kesinti önleme statü bitini bir yap
STA Akümülatördeki değeri belleğe sakla
STX X indeksini belleğe sakla
STY Y indeksini belleğe sakla
TAX Akümülatörün değerini X indeksine gönder
TAY Akümülatörün değerini Y indeksine gönder
TSX Yığın Göstergecini X indeksine gönder
TXA X İndeksini akümülatöre gönder
TXS X İndeksini yığın kaydına gönder
TYA Y indeksini akümülatöre gönder

PROGRAMLAMA MODELİ



6510 BELLEK HARİTASI



UYGULAMA NOTLARI

Çıkış kaydını sıfırıncı sayfadaki iç Giriş-Çıkış portuna yerleştirmek. 6510'un güçlü sıfırıncı sayfa adresleme komutlarını Giriş-Çıkış uçlarını giriş olarak tanımlayarak (Veri Yön Kaydını kullanarak) kullanıcı, 0001 adresinin (Çıkış Kaydı) içeriğini çevre birim cihazlarını kullanarak değiştirebilme yetisine kavuşacaktır. Çevre birimlerinin girişlerini kullanarak bu içeriklerin değiştirilmesi yetisi, sıfırıncı sayfa Dolaylı Adresleme komutları ile birlikte, yeni ve çok yönlü programlama tekniklerinin geliştirilmesine yardımcı olacaktır.

NOT: COMMODORE SEMICONDUCTOR GROUP, güvenilirliği, işlevi veya tasarımı iyileştirmek amacıyla burada yer alan herhangi bir üründe değişiklik yapma hakkını saklı tutar. COMMODORE SEMICONDUCTOR GROUP, burada açıklanan herhangi bir ürün veya devrenin uygulanması veya kullanımından kaynaklanan hiçbir sorumluluğu kabul etmez; ayrıca kendi patent hakları veya başkalarının hakları kapsamında herhangi bir lisans vermez.

INSTRUCTIONS		Immediate			Absolute			Zero Page			Accum.			Implied			(Ind.) X			(Ind.) Y			Z, Page, X			Abs. X			Abs. Y			Relative			Indirect			Z, Page, Y			CONDITION CODES					
Mnemonic	Operation	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
ADC	A + M + C → A (4) (1)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										✓	✓	✓	–	–	✓
AND	A Λ M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										✓	✓	–	–	–	–
ASL	C ← [7 0] ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													✓	✓	✓	–	–	–
BCC	BRANCH ON C=0 (2)																															90	2	2							–	–	–	–	–	–
BCS	BRANCH ON C=1 (2)																															B0	2	2							–	–	–	–	–	–
BEO	BRANCH ON Z=1 (2)																															F0	2	2							–	–	–	–	–	–
BIT	A Λ M				2C	4	3	24	3	2																															M_7	✓	–	–	–	M_6
BMI	BRANCH ON N=1 (2)																															30	2	2							–	–	–	–	–	–
BNE	BRANCH ON Z=0 (2)																															D0	2	2							–	–	–	–	–	–
BPL	BRANCH ON N=0 (2)																															10	2	2							–	–	–	–	–	–
BRK	(See Fig. 1)													00	7	1																									–	–	–	1	–	–
BVC	BRANCH ON V=0 (2)																															50	2	2							–	–	–	–	–	–
BVS	BRANCH ON V=1 (2)																															70	2	2							–	–	–	–	–	–
CLC	0 → C													18	2	1																									–	–	0	–	–	–
CLD	0 → D													D8	2	1																									–	–	–	–	0	–
CLI	0 → I													58	2	1																									–	–	–	0	–	–
CLV	0 → V													B8	2	1																									–	–	–	–	–	0
CMP	A - M (1)	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										✓	✓	✓	–	–	–
CPX	X - M	E0	2	2	EC	4	3	E4	3	2																															✓	✓	✓	–	–	–
CPY	Y - M	C0	2	2	CC	4	3	C4	3	2																															✓	✓	✓	–	–	–
DEC	M - 1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													✓	✓	–	–	–	–
DEX	X - 1 → X													CA	2	1																									✓	✓	–	–	–	–
DEY	Y - 1 → Y													B8	2	1																									✓	✓	–	–	–	–
EOR	A ⊻ M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										✓	✓	–	–	–	–
INC	M + 1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													✓	✓	–	–	–	–
INX	X + 1 → X													E8	2	1																									✓	✓	–	–	–	–
INY	Y + 1 → Y													C8	2	1																									✓	✓	–	–	–	–
JMP	JUMP TO NEW LOC.				4C	3	3																												6C	5	3				–	–	–	–	–	–
JSR	(See Fig. 2) JUMP SUB				20	6	3																																		–	–	–	–	–	–
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										✓	✓	–	–	–	–

ZAMAN, BELLEK GEREKSİNİMLERİ

INSTRUCTIONS		Immediate			Absolute			Zero Page			Accum.			Implied			(Ind.) X			(Ind.) Y			Z, Page, X			Abs. X			Abs. Y			Relative			Indirect			Z, Page, Y			CONDITION CODES
Mnemonic	Operation	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N Z C I D V
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3							B6	4	2	✓ ✓ – – – –
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													✓ ✓ – – – –
LSR	0 → [7 0] → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0 ✓ ✓ – – –
NOP	NO OPERATION													EA	2	1																									– – – – – –
ORA	A V M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										✓ ✓ – – – –
PHA	A → MS S - 1 → S													48	3	1																									– – – – – –
PHP	P → MS S - 1 → S													08	3	1																									– – – – – –
PLA	S + 1 → S M_S → A													68	4	1																									✓ ✓ – – – –
PLP	S + 1 → S M_S → P													28	4	1																									(RESTORED)
ROL	←[7 0]←[C]←				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													✓ ✓ ✓ – – –
ROR	→[C]→[7 0]→				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													✓ ✓ ✓ – – –
RTI	(See Fig. 1) RTRN INT													40	6	1																									(RESTORED)
RTS	(See Fig. 2) RTRN SUB													60	6	1																									– – – – – –
SBC	A - M - $\bar{C}$ → A (1)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										✓ ✓ (3) – – ✓
SEC	1 → C													38	2	1																									– – (1) – – –
SED	1 → D													F8	2	1																									– – – – (1) –
SEI	1 → I													78	2	1																									– – – (1) – –
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3										– – – – – –
STX	X → M				8E	4	3	86	3	2																												96	4	2	– – – – – –
STY	Y → M				8C	4	3	84	3	2													94	4	2																– – – – – –
TAX	A → X													AA	2	1																									✓ ✓ – – – –
TAY	A → Y													AB	2	1																									✓ ✓ – – – –
TSX	S → X													BA	2	1																									✓ ✓ – – – –
TXA	X → A													8A	2	1																									✓ ✓ – – – –
TXS	X → S													9A	2	1																									– – – – – –
TYA	Y → A													98	2	1																									✓ ✓ – – – –

(1) SAYFA SINIRI GEÇİLİRSE "N"YE 1 EKLE
(2) DALLANMA AYNI SAYFAYA OLURSA "N"YE 1 EKLE
BAŞKA SAYFAYA OLURSA 2 EKLE
(3) TAŞIMAYIN = ÖDÜNÇ ALIN
(4) N ONDALIK MOD Z BAYRAĞI GEÇERSİZ İSE
TOPLAYICI SIFIR SONUÇ İÇİN KONTROL EDİLMELİDİR.

X İNDEKS X
Y İNDEKS Y
A AKÜMÜLATÖR
M ETKİLİ ADRES BAŞINA BELLEK
M_S YIĞIN İŞARETİ BAŞINA BELLEK

\+ TOPLA
\- ÇIKAR
Λ VE
V VEYA
v ÖZEL VEYA

√ DEĞİŞTİRİLMİŞ
\- DEĞİŞTİRİLMEMİŞ
M7 BELLEK BİTİ 7
M6 BELLEK BİTİ 6
N DÖNGÜ SAYISI
\# BAYT SAYISI

NOT: COMMODORE SEMICONOUCTOR GROUP, tanımlanmamış OP KODLARIN kullanılmasını pek tavsiye etmez ve güvence vermez.

EK M

6526 KARMAŞIK ARAYÜZ ADAPTÖRÜ (CIA) ÇİPİ

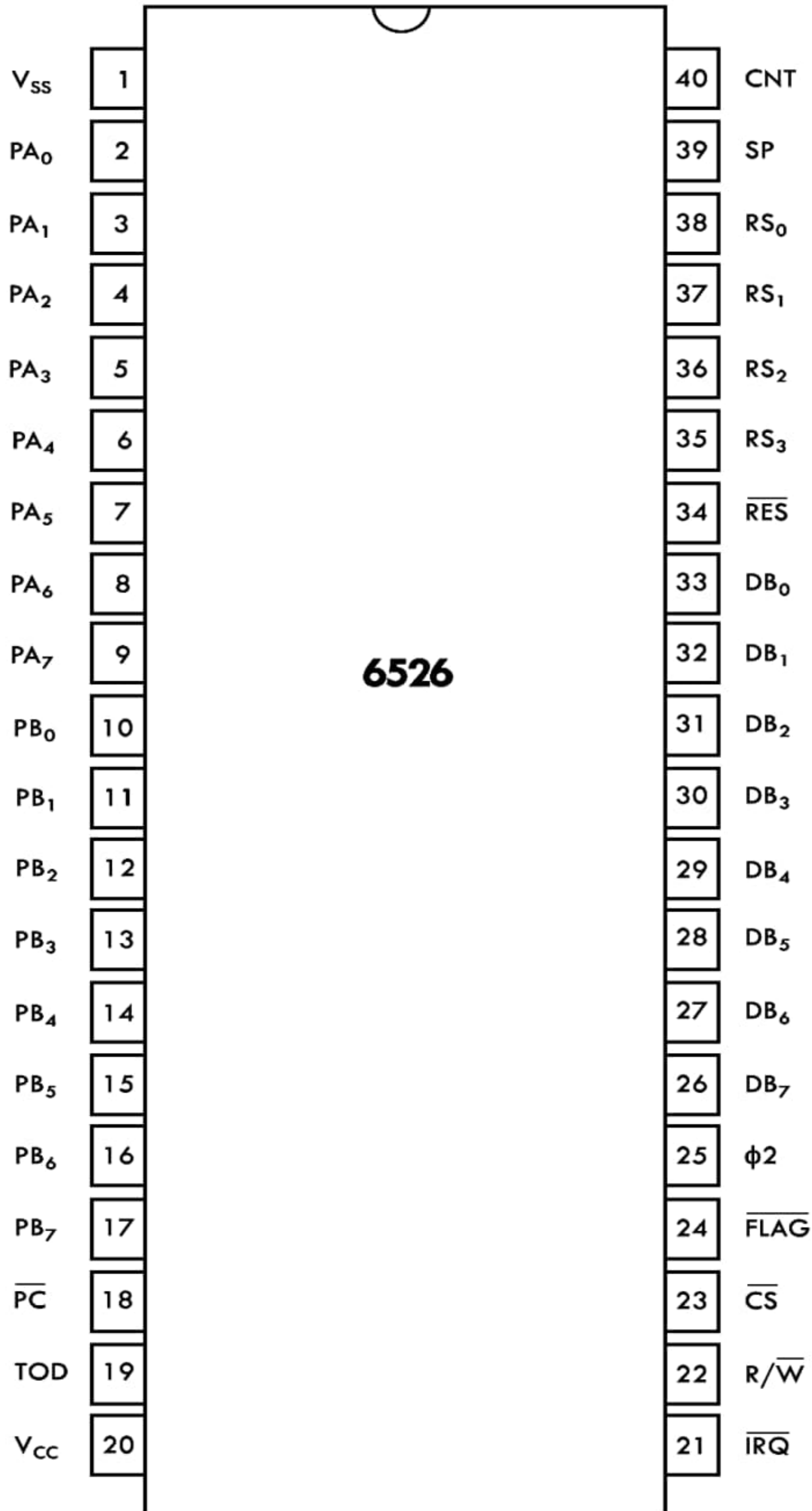
TANIMLAMA

6526 CIA, çok esnek zamanlamaya ve Giriş/Çıkış özelliklerine sahip, 65XX veri yolu uyumlu, dış arabirim cihazıdır.

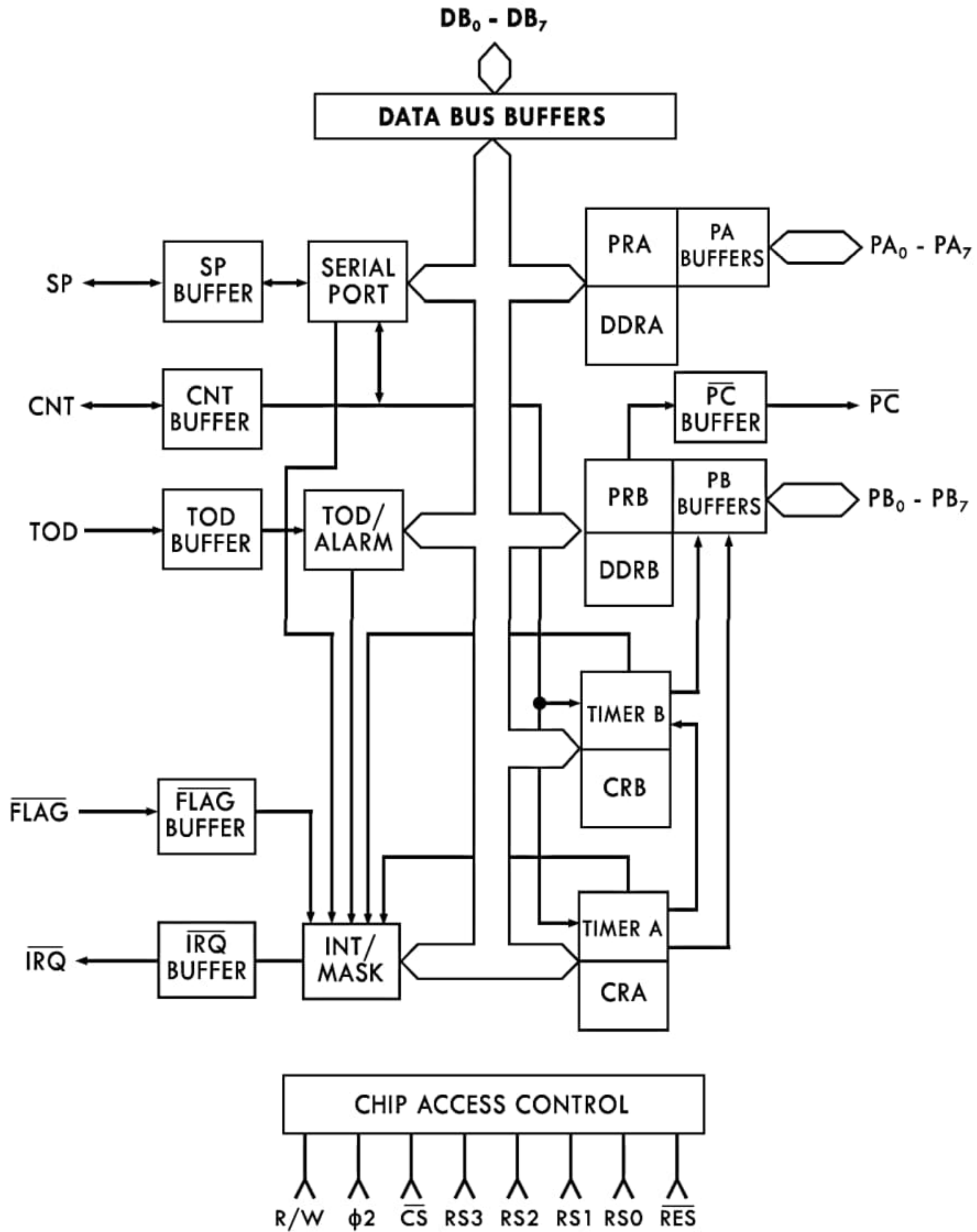
ÖZELLİKLERİ

- Tek tek programlanabilir 16 Giriş/Çıkış hattı
- Okuma veya yazma için, 8 ya da 16 bitlik El sıkışma turu haberleşme
- 2 bağımsı, bağlantı kurulabilir 16 bitlik ara zamanlayıcı
- Programlanabilir alarmlı 24-saatlik günlük zaman saati
- Seri Giriş/Çıkış için 8 bittik kayma kaydı
- 2TTL yükleme özelliği
- CMOS uyumlu Giriş/Çıkış hatları
- 1 veya 2 MHz ile çalışabilme olanağı

6526 BACAK YAPILANDIRMASI



6526 BLOK DİYAGRAMI



MAKSİMUM ÇALIŞMA DURUMLARI

Voltaj Kaynağı, V_{CC}	-0.3V'dan + 7.0V'a kadar
Giriş/Çıkış Voltajı, V_{IN}	-0.3V'dan + 7.0V'a kadar
Çalışma Isısı, T_{OP}	0°C'dan 70°C'ye kadar
Saklanım Isısı, T_{STG}	-55°C'den 150°C'ye kadar

Tüm girişler, yüksek statik elektrik boşalımına karşı koruma devresine sahiptir. Verilen sınırları aşan gereksiz voltaj uygulamalarına karşı çok dikkatli olunmalıdır.

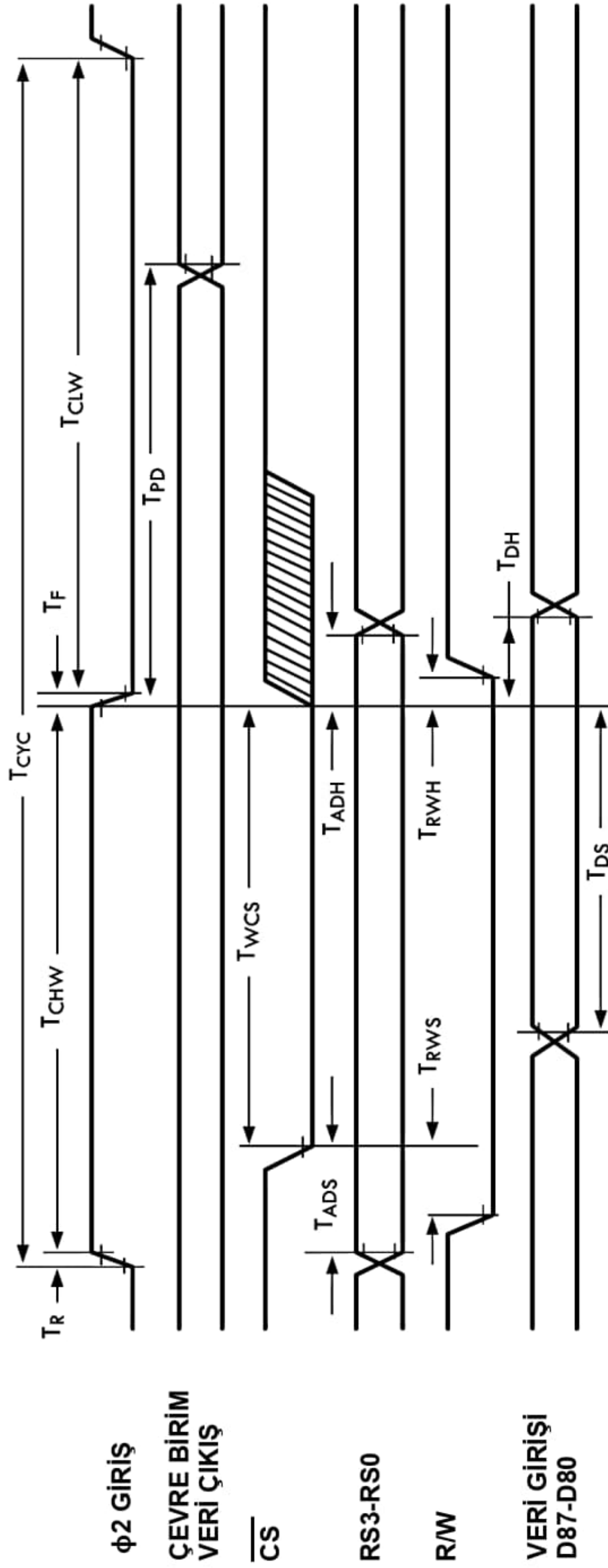
UYARILAR

Cihazın tahammül gücünü gösteren maksimum çalışma durumlarından yüksek değerler kalıcı bozukluklar oluşturabilir, hatta bu durumlara yakın çalışma hallerinde bile, sürenin uzaması cihazın güvenirliğini etkileyebilir.

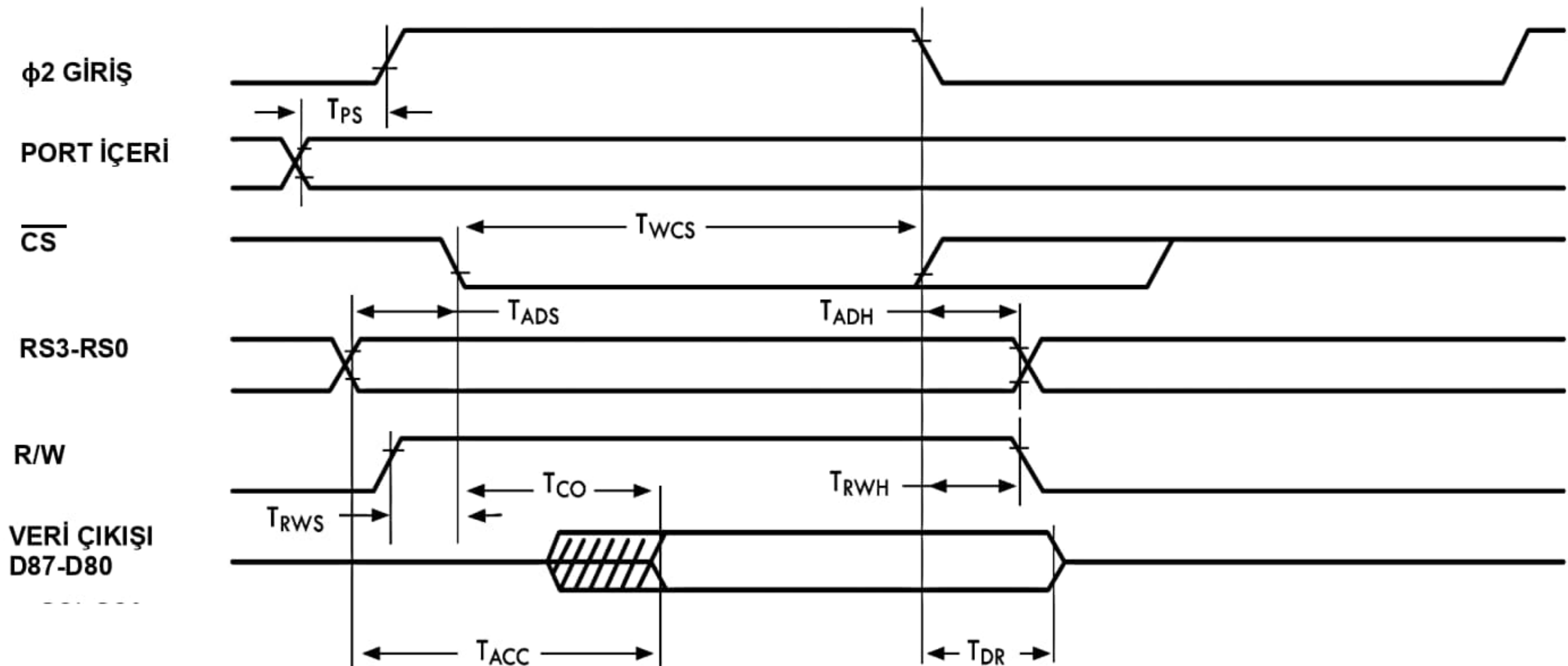
ELEKTRİKSEL KARAKTERİSTİKLER (V_{CC} ± 5%, V_{SS} = 0 V, T_A = 0-70°C)

KARAKTERİSTİK	SEMBOL	MİN.	TİP	MAK.	BİRİM
Yüksek Voltaj Girişi	V_{IH}	+2.4	—	V_{CC}	V
Düşük Voltaj Girişi	V_{IL}	-0.3	—	—	V
Giriş Kaçak Akımı; V_{IN}=V_{SS} + 5V (TOD, R/W, $\overline{FLAG}$, ϕ2, $\overline{RES}$, RS0-RS3, $\overline{CS}$	I_{IN}	—	1.0	2.5	μA
Port Giriş Çekme Direnci	R_{PI}	3.1	5.0	—	KΩ
Yüksek Empedans Durumu (Üç Durum) için Çıkış Kaçak Akımı; V_{IN} = 4V'dan 2.4V'a; (DB0—DB7, SP, CNT, $\overline{IRQ}$)	I_{TSI}	—	±1.0	±10.0	μA
Yüksek Voltaj Çıkışı V_{CC} = MIN, I_{LOAD} < -200 μA (PA0—PA7, $\overline{PC}$ PB0—PB7, DB0—DB7)	V_{OH}	+2.4	—	V_{CC}	V
Düşük Voltaj Çıkışı V_{CC} = MIN, I_{LOAD}, < 3.2 mA	V_{OL}	—	—	+0.40	V
Çıkış Yüksek Akım (Kaynak); V_{OH} > 2.4V (PA0—PA7, PB0—PB7, $\overline{PC}$, DB0—DB7)	I_{OH}	-200	-1000	—	μA
Çıkış Düşük Akımı (Batan); V_{OL} < .4V (PA0—PA7, $\overline{PC}$, PB0—PB7, DB0—DB7)	I_{OL}	3.2	—	—	mA
Giriş Kapasitansı	C_{IN}	—	7	10	pf
Çıkış Kapasitansı	C_{OUT}	—	7	10	pf
Güç Kaynağı Akımı	I_{CC}	—	70	100	mA

6526 YAZMA ZAMANLAMA DİYAGRAMI



6526 OKUMA ZAMANLAMA DİYAGRAMI



6526 ARABİRİM SİNYALLERİ

ϕ2-SAAT GİRİŞİ

•2 saati, cihaz içi işlemler ve sistem veri yolu ile haberleşmek için zamanlama referansı kullanılan, TTL uyumlu bir giriştir.

$\overline{CS}$-ÇİP SEÇİM GİRİŞİ

$\overline{CS}$ girişi 6526'nın işlemlerini kontrol eder. •2 yüksek durumdayken $\overline{CS}$'ın alçak seviyesinde alması, cihazın R/W ve adres (RS) hatlarına karşılık vermesine sebep olur. $\overline{CS}$'in yüksek olması, bu hatların 6526'yı kontrol edememesine neden olur. $\overline{CS}$ hattı normalde uygun adres birleşimleri ile •2'de işler duruma (alçak) geçer.

R/W-OKUMA/YAZMA GİRİŞİ

R/W sinyali mikroişlemci tarafından sağlanır ve 6526'nın veri transferinin yönünü kontrol eder. R/W girişinin yüksek olması, okuma işlemini (6526'dan dışarıya veri transferi), alçak olması yazma işlemini (6526'ya veri transferi) gösterir.

AS3-RS0-Adres Girişleri

Adres girişleri, Kayıt Haritasında açıklanan, iç kayıtların seçiminde kullanılır.

DB7 BD0-Veri Yolu Giriş/Çıkışları

8 Veri Yol ucu. 6526 ile sistem veri yolu arasında bilgi transfer etmek amacıyla kullanılır. Bu uçlar, okuma durumunda, $\overline{CS}$ alçak, R/W ve •2 yüksek olmadıkça, empedanslı girişlerdir. Okuma sırasında, veri yolu çıkış tamponları, seçilen kayıttan sistem veri yoluna bilgi sürecek şekilde, çalışma durumuna geçerler.

$\overline{IRQ}$-Kesinti isteği Çıkışı

$\overline{IRQ}$, normalde işlemci kesinti girişine bağlı, açık "drain" çıkışıdır. Bir dış çekme (pull-up) direnci, birden fazla $\overline{IRQ}$ çıkışının beraber bağlanmasını sağlayacak şekilde sinyali yüksek düzeyde tutar. $\overline{IRQ}$ çıkışı normalde, yüksek dirençlidir, alçak düzeyde çalıştırılması, işlevsel tanımlamada gösterilmiştir.

$\overline{RES}$-Reset Girişi

$\overline{RES}$ ucunun alçak duruma geçmesi, tüm iç kayıtların silinmesine (reset) sebep olur. Port uçları, giriş olarak seçilir. Port ve zamanlama kontrol kayıtları o değerini alır. Diğer tüm kayıtlar da sıfırlanır.

6526 ZAMANLAMA KARAKTERİSTİKLERİ

Sembol	Karakteristik	1 MHz		2 MHz		Birim
		MİN.	MAK.	MİN.	MAK.	
	ϕ2 Saat					
T_{CYC}	Döngü Süresi	1000	20,000	500	20,000	ns
T_R, T_F	Yükseliş ve Düşüş Zamanı	—	25	—	25	ns
T_{CHW}	Saat Darbe Genişliği (Yüksek)	420	10,000	200	10,000	ns
T_{CLW}	Saat Darbe Genişliği (Düşük)	420	10,000	200	10,000	ns
	Yazma Döngüsü					
T_{PD}	•2'den Çıkış Gecikmesi	—	1000	—	500	ns
T_{WCS}	$\overline{CS}$ Düşükken •2 Yüksek	420	—	200	—	ns
T_{ADS}	Adres Kurulum Süresi	0	—	0	—	ns
T_{ADH}	Adres Bekletme Süresi	10	—	5	—	ns
T_{RWS}	R/W Kurulum Süresi	0	—	0	—	ns
T_{RWH}	R/W Bekletme Süresi	0	—	0	—	ns
T_{DS}	Veri Yolu Kurulum Süresi	150	—	75	—	ns
T_{DH}	Veri Yolu Bekletme Süresi	0	—	0	—	ns
	Okuma Döngüsü					
T_{PS}	Port Kurulum Süresi	300	—	150	—	ns
T_{WCS}(2)	$\overline{CS}$ Düşükken •2 Yüksek	420	—	20	—	ns
T_{ADS}	Adres Kurulum Süresi	0	—	0	—	ns
T_{ADH}	Adres Bekletme Süresi	10	—	5	—	ns
T_{RWS}	R/W Kurulum Süresi	0	—	0	—	ns
T_{RWH}	R/W Bekletme Süresi	0	—	0	—	ns
T_{ACC}	RS3-RS0'dan Veri Erişimi	—	550	—	275	ns
T_{CO}(3)	$\overline{CS}$'den Veri Erişimi	—	320	—	150	ns
T_{DR}	Veri Yayın Süresi	50	—	25	—	ns

NOT:
1-Tüm zamanlamalar, girişlerde V_{IL} maksimum V_{IH} minimum, çıkışlarda V_{OL} maksimum ve V_{OH} minimumdan referans edilirler.
2-T_{WCS} en son yüksek •2 veya alçak $\overline{CS}$'den ölçülür. $\overline{CS}$'nin, •2 yüksek durumunun sonuna kadar alçak olarak tutulması gerekir.
3-T_{CO} en son yüksek •2 veya alçak $\overline{CS}$'den olçüıür. Sadece son T_{ACC} veya T_{CO}'dan sonra geçerli veri mümkündür.

KAYIT HARİTASI

RS3	RS2	RS1	RS0	KAYIT	İSİM	
0	0	0	0	0	PRA	Çevre birim veri kaydı A
0	0	0	1	1	PRB	Çevre birim veri kaydı B
0	0	1	0	2	DDRA	Veri yön kaydı A
0	0	1	1	3	DDRB	Veri yön kaydı B
0	1	0	0	4	TA LO	A zamanlayıcı düşük kaydı
0	1	0	1	5	TA HI	A zamanlayıcı yüksek kaydı
0	1	1	0	6	TB LO	B zamanlayıcı düşük kaydı
0	1	1	1	7	TB HI	B zamanlayıcı yüksek kaydı
1	0	0	0	8	TOD 10THS	10'lu saniye kaydı
1	0	0	1	9	TOD SEC	Saniye kaydı
1	0	1	0	A	TOD MIN	Dakika kaydı
1	0	1	1	B	TOD HR	Saat – AM/PM kaydı
1	1	0	0	C	SDR	Seri veri kaydı
1	1	0	1	D	ICR	Kesinti kontrol kaydı
1	1	1	0	E	CRA	Kontrol kaydı A
1	1	1	1	F	CRB	Kontrol kaydı B

6526 İŞLEVLERİ

Giriş/Çıkış Portları (PRA, PRB, DDRA, DDRB)

Hem Port A hem de Port B, 8 bitlik bir Çevre-Birim Veri Kaydı (PR), bir de 8 bitlik Veri Yön Kaydından (DDR) oluşmuştur. Eğer DDR'daki bir bit "1" ise, PR'de bu bite karşılık gelen bit çıkış, eğer DDR'deki bir bit "0" ise, PR'de bu bite karşılık gelen bit giriş olarak tanımlanır. READ (okuma) sırasında PR hem giriş hem de çıkış bitleri için, gerçek port uçları (PA0-PA7-PB0-PB7) üzerindeki bilgiyi yansıtır. Port A ve Port B, CMOS ve TTL'e uygun, aktif ve pasif pull-up cihazlarıdır. Her iki portta da iki TTL load drive yetisi vardır. Normal giriş/çıkış işlemlerine ek olarak, PB6 ve PB7 zamanlayıcı çıkış fonksiyonlarını da sağlarlar.

El Şıkışma (Handshaking)

Veri transferlerinde, el sıkışma türü haberleşme. $\overline{PC}$ çıkış ucu ve $\overline{FLAG}$ giriş ucu kullanılarak gerçekleştirilir. Port B'nin okuma ve yazmasından sonra $\overline{PC}$, bir çevrim için alçak düzeye geçecektir. Bu sinyal, PORT B'de verinin hazır olduğunu ("veri hazır") ya da PORT B'nin veriyi kabul ettiğini ("veri kabul edildi") göstermek için kullanılır. 16-bitlik veri transferlerinde (hem Port A'yı hem de Port B'yi kullanarak) el sıkışma, genellikle ilk olarak Port A'nın okuma ve yazması ile mümkün olabilir. $\overline{FLAG}$ negatif kenara duyarlı bir giriştir ve genel amaçlı kesinti girişi olarak ya da diğer bir 6526'dan $\overline{PC}$ çıkışı almak için kullanılabilir. $\overline{FLAG}$'in herhangi bir negatif geçişi ile $\overline{FLAG}$ kesinti biti set edilir.

KAYIT	İSİM	D7	D6	D5	D4	D3	D2	D1	D0
0	PRA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
1	PRB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
2	DDRA	DPA7	DPA6	DPA5	DPA4	DPA3	DPA2	DPA1	DPA0
3	DDRB	DPB7	DPB6	DPB5	DPB4	DPB3	DPB2	DPB1	DPB0

Aralık Zamanlayıcıları (A Zamanlayıcısı, B Zamanlayıcısı)

Her bir aralık zamanlayıcısı 16-bitlik yalnız-okunabilen bir zamanlayıcı sayacından ve 16-bitlik yalnız-yazılabilir zamanlayıcı Latch'inden oluşmuştur. Zamanlayıcıdan okunan veri, zamanlayıcı sayacın o andaki içeriğidir. Zamanlayıcıya yazılmış olan veri ise zamanlayıcı Latch'inde tutulur. Zamanlayıcılar, daha kapsamlı işler için birlikte ya da birbirinden bağımsız olarak kullanılabilirler. Çeşitli zamanlayıcı modları; uzun süreli beklemelerin, değişken genişlikli vuruşların, vuruş dizilerinin ve değişik frekanslı dalga biçimlerinin oluşturulmasını sağlarlar. CNT girişinin kullanılması ile, zamanlayıcılar dış vuruşlar; sayabilir ya da dış sinyalli gecikme zamanlarını, vuruş genişliğini ve frekansını ölçebilirler. Her zamanlayıcının, aşağıdaki fonksiyonların bağımsız olarak kontrol edilmesini sağlayan birer kontrol kaydı vardır:

Başlatma/Durdurma

Bir kontrol biti zamanlayıcının, mikroişlemciyi herhangi bir anda başlatmasını ya da durdurmasını sağlar.

PB Açık/Kapalı

Bir kontrol biti zamanlayıcı çıkışının, PORT B'nin bir çıkış hattında görünmesini sağlar (Zamanlayıcı A için PB6 ve Zamanlayıcı B için PB7). Bu fonksiyon DDRB kontrol bitini devreden çıkarır ve uygun PB hattını bir çıkış için zorlar.

Vuruş (Toggle)

Bir kontrol biti, PORT B'ye uygulanan çıkışın seçimini yapar. Bir döngü sürelik pozitif vurumu toggle eder veya üretir. Toggle çıkışı, zamanlayıcı çalışmaya başladığında yüksek düzeye gelir ve sadece $\overline{RES}$ ile alçak olur.

Tek Atış/Sürekli (One Shot/Continucus)

Bir kontrol biti zamanlayıcı modlarını seçer. Tek atış modunda zamanlayıcı; tutulan değerden sıfıra kadar geri sayar, bir kesinti üretir, tutulan değeri tekrar yükler ve sonra durur. Sürekli modda zamanlayıcı, tutulan değerden sıfıra kadar sayacak, bir kesinti üretecek, tutulan değeri tekrar yükleyecek ve bu işlemi sürekli tekrarlayacaktır.

Kuvvet Yükü (Force Load)

Bir tekrarlı uyarı (strobe) biti, zamanlayıcı Latch'inin herhangi bir anda zamanlayıcı sayacına yüklenmesini sağlar. Bu işlem, zamanlayıcının çalışıp çalışmadığına bağlı değildir.

Giriş Modu

Kontrol bitleri, zamanlayıcıyı azaltmak için saat seçimi amacıyla kullanılır. A zamanlayıcısı ya da •2 saat vuruşlarını CNT ucuna uygulanan dış vurumları sayar. B zamanlayıcısı ise •2 saat ve dış CNT vurumlarını A zamanlayıcısı "undeıilow" vurumları veya CNT ucu high durumdayken A zamanlayıcısı '"underflow2" vurumlarını sayar. Herhangi bir zamanlama latch'i zamanlayıcıya taşma, force load veya zamanlayıcı durduktan sonra "prescaler" yüksek baytına yazım işleminden sonra yüklenir. Zamanlayıcı çalışıyorsa, yüksek bayta yazım, zamanlama latch'ine yükleme yapar fakat sayaca tekrar yükleme işlemi oluşmaz.

OKUMA (ZAMANLAYICI)
KAYIT İSİM

4	TA LO	TAL_7	TAL_6	TAL_5	TAL_4	TAL_3	TAL_2	TAL_1	TAL_0
5	TA HI	TAH_7	TAH_6	TAH5	TAH_4	TAH_3	TAH_2	TAH_1	TAH_0
6	TB LO	TBL_7	TBL_6	TBL_5	TBL_4	TBL_3	TBL_2	TBL_1	TBL_0
7	TB HI	TBH_7	TBH_6	TBH_5	TBH_4	TBH_3	TBH_2	TBH_1	TBH_0

YAZMA (ÖN ÖLÇEKLEYİCİ)
KAYIT İSİM

4	TA LO	PAL_7	PAL_6	PAL_5	PAL_4	PAL_3	PAL_2	PAL_1	PAL_0
5	TA HI	PAH_7	PAH_6	PAH_5	PAH_4	PAH_3	PAH_2	PAH_1	PAH_0
6	TB LO	PBL_7	PBL_6	PBL_5	PBL_4	PBL_3	PBL_2	PBL_1	PBL_0
7	TB HI	PBH_7	PBH_6	PBH_5	PBH_4	PBH_3	PBH_2	PBH_1	PBH_0

GÜNLÜK ZAMAN SAATİ (TOD: Time of day clock)

TOD saati, gerçek zaman uygulamaları için kullanılan özel amaçlı bir zamanlayıcıdır. TOD, 1/10 saniye duyarlıklı bir 24 saatlik (AM-PM) saati içerir. 4 kayıtlı olarak planlanmıştır. Bu kayıtların içindeki değerler, saniyenin 10'da biri, saniye, dakika ve saattir. AM/PM bayrağı, kolay bit testi için saat kaydının en saldaki bitidir. Her kayıt, ekran görüntüsü yaratma işlemi için, değiştirme işlemini kolaylaştırmak amacıyla BCD (ikili kodlanmış biçimi) ile okunur. Saatin, zamanı doğru olarak takip edebilmesi için, TOD ucu üzerinde, dıştan bir 60 veya 50 Hz TTL seviyesine (programlanabilir) gereksinimi vardır. Zaman takibine ek olarak, istenilen zamanda kesim işlemini elde etmek amacıyla programlanabilir bir ALARM işlemi sağlanmıştır. Alarm kayıtları, karşı gelen TOD kayıtlarıyla aynı adreslere yerleştirilmişlerdir. ALARMA erişim, bir Kontrol kaydı bitiyle sağlanır. ALARM salt-yazılabilir özelliktedir ve TOD adresini okumak için yapılacak bir işlem sonucunda. ALARM erişim biti durumundan bağımsız olarak, zaman okunacaktır.

Uygun TOD okuma ve yerleştirme için, bir dizi özel olayın takip edilmesi gerekir. TOD saat kaydına yapılan bir yazma işleminden sonra, otomatik olarak durur ve saniyenin 10'da biri kaydına yapılan bir yazma işleminden önce çalışmaz. Bu TOD'un istenilen zamanda başlamasını sağlar. Okuma işlemine bağlı olarak, bir basamaktan diğerine elde işlemi yapılabileceğinden, okuma süresince, günlük zaman bilgisinin sabit tutulması amacıyla, latching işlemine gerek duyulur. Dört TOD kayıtlarının tümü, saatin okunması ile latch edilir ve saniyenin onda biri kaydı okununcaya kadar bu durumunda kalır. TOD saati, çıkış kayıtları latch edildiğinde, saymaya devam eder. Tek bir kayıt okunacaksa, elde problemi olmaz ve kayıt değeri hemen okunabilir. Ancak latching işleminin durdurulması amacıyla, saat kaydının okunmasından sonra, saniyenin onda biri kaydının okunması da gereklidir.

OKUMA
KAYIT İSİM

8	TOD 10THS	0	0	0	0	T_8	T_4	T_2	T_1
9	TOD SEC	0	SH_4	SH_2	SH_1	SL_8	SL_4	SL_2	SL_1
A	TOD MIN	0	MH_4	MH_2	MH_1	ML_8	ML_4	ML_2	ML_1
B	TOD HR	PM	0	0	HH	HL_8	HL_4	HL_2	HL_1

YAZMA

CRB7 = 0 TOD
CRB7 = 1 ALARM
(OKUMA İLE AYNI BİÇİMDE)

Seri Port (SDR)

Seri port, tamponlanmış 8-bitlik eş zamanlı kayma kaydı sistemidir. Bir kontrol biti, giriş veya çıkış modunu seçer. Giriş modunda, SP ucundaki veri, CNT ucuna uygulanan sinyalin yükselme anında, kayma kaydına aktarılır. 8 CNT vuruşundan sonra, kayma kaydındaki veri, seri Veri Kaydına gönderilir ve bir kesinti üretilir. Çıkış modunda, A zamanlayıcısının baud hızı üretici olarak kullanılır. Veri, SP ucunda, A zamanlayıcısının underflow hızının 1/2'si ile dışarı aktarılır. Mümkün olabilecek en büyük baud hızı. •2 bölü 4'tür, fakat kullanılabilen maksimum baud hızı, hattın yük durumu ve alıcının cevap verme hızı ile belirlenir. Gönderme işlemi, Seri Veri Kaydına yapılacak bir yazma işlemi ile başlar <A zamanlayıcısının çalışması ve sürekli modda olması gerekir). A zamanlayıcısından elde edilen saat sinyali, CNT ucunda çıkış olarak belirir. Seri veri kaydındaki veri, kayma kaydına yüklendikten sonra, CNT vuruşu oluştuğunda SP ucuna aktarılacaktır. Aktarılan veri. CNT sinyalinin düşme kısmında geçerli olur ve bir sonraki düşme kısmına kadar geçerli olarak kalır. 8 CNT vuruşundan sonra, daha fazla veri gönderilebileceğini gösteren bir kesinti yaratır. Bu kesintiden önce, seri veri kaydı, yeni bilgi ile yüklenirse, yeni veri otomatik olarak kayma kaydına yüklenir ve gönderme işlemi devam eder. Eğer mikroişlemci: kayma kaydından bir bayt öndeyse, bu işlem sürer. Başka bilgi gönderilmeyecekse, 8'inci CNT vuruşundan sonra. CNT yüksek olur ve SP. son gönderilen veri bitin seviyesinde kalır. SDR verisi, en soldaki biti önce olacak şekilde kaydırılır ve seri giriş verisi de aynı bu şekilde alınır.

Seri Port ve CNT saatinin çift yönlülük özelliği, birçok 6526 cihazının genel bir seri haberleşme veri yoluna bağlanabilmesini sağlar. Burada bir 6526, veri kaynağı ve kayma saatini içeren şef (master) konumunda, diğer 6526 çipleri ise yardımcıları (slave) olarak çalışırlar. CNT ve SP çıkışlarının her ikisi de böyle genel bir veri yoluna izin verecek şekilde "açık kanal” özelliği taşırlar. Şef/Yardımcı seçimi için protokol, seri veri yolu üzerinde veya belirtilen el sıkışma hatlarında iletilir.

KAYIT İSİM

C	SDR	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

Kesinti Kontrolü (ICR)

6526'da beş kesinti bayrağı (interrupt) vardır; A zamanlayıcının underflow'u, B zamanlayıcısının underflow'u, TOD ALARM 1, Seri Port dolu/boş sinyali ve $\overline{\text{FLAG}}$, Tek bir kayıt maskeleme ve kesinti bilgisi sağlar. Kesinti Kontrol Kaydı, yalnızca-yazılabilir MASK kaydı ile yalnızca-okunabilir VERİ kaydına sahiptir. Herhangi bir kesinti işlemi, veri kaydında, karşılık gelen biti "1" yapar. MASK kaydı tarafından çalışmasına izin verilen bir kesinti. VERİ kaydının IR (MSB: en soldaki bit) bitini bir seviyesine ve $\overline{\text{IRQ}}$ ucunu da sıfır seviyesine getirir Çok-çipli sistemlerde IR biti, kesinti isteğini hangi çipin gönderdiğini anlamak amacıyla sürekli yüklenir. VERİ kaydının okunmasından sonra, kesinti VERİ kaydı sıfırlanır ve $\overline{\text{IRQ}}$ hattı yüksek duruma geçer. Her kesinti, MASK'ı önemsemeden bir kesinti bitini 1 yaptığından ve bir işlemci kesintisinin oluşmasını önlemek için, her kesinti biti seçime bağlı olarak maskelendiğinden, doğru kesintiler ile kaydedilmiş kesintileri birbirine karıştırmak mümkündür. Buna karşılık IR bitinin sürekli yoklanması, VERİ kaydının sıfırlanmasına sebep olur, böylece herhangi bir kesinti işlemi sırasında kullanıcının, VERİ kaydındaki bilgiyi koruması gerekir.

MASK (maskeleme) kaydı, maskeleme bitlerinin tek tek, uygun bir şekilde kontrol edilmesini sağlar. Maskeleme kaydına yazma işlemi uygulanırken, verinin 1'nci biti (SET/$\overline{\text{CLEAR}}$) sıfırsa, bir olan tüm maskeleme bitleri sıfırlanır. Buna karşılık 0 olan bitler bu işlemden etkilenmez. Verinin 7'nci biti bir ise, bir olan tüm maskeleme bitleri set edilir, diğerleri olduğu gibi kalır. IR'yi set etmek gereken bir kesinti bayrağı ve kesinti isteği sinyali üretmek için, karşılık gelen MASK bitinin set edilmesi gerekir.

OKUMA (INT DATA)

KAYIT İSİM

D	ICR	IR	0	0	FLG	SP	ALRM	TB	TA

YAZMA (INT MASK)

KAYIT İSİM

D	ICR	S/$\overline{\text{C}}$	X	X	FLG	SP	ALRM	TB	TA

KONTROL KAYITLARI

6526'da, CRA ve CRB olarak bilinen iki kontrol kaydı vardır. CRA, A zamanlayıcısı, CRB ise B zamanlayıcısı ile çalışır. Kayıt biçimi aşağıda verilmiştir:

CRA:

Bit	İsim	İşlevi
0	START	1 = A Zamanlayıcısını başlatır. 1 = A Zamanlayıcısını durdurur. Tek-atış modu sırasında underflow oluşursa bu bit otomatik olarak reset edilir.
1	PBON	1 = A Zamanlayıcısı PB6 üzerinde görünür. 0 = PB6 normal işlemini görür.
2	OUTMODE	1 = Geçiş, 0 = Vuruş
3	RUNMODE	1 = Tek-Atış, 0 = Sürekli
4	LOAD	1 = Kuvvet Yükü (Bu STROBE girişidir, veriler saklanmaz, 4'uncu bit geriye her zaman sıfır okuyacaktır ve sıfır yazmanın hiçbir etkisi yoktur).
5	INMODE	1 = A Zamanlayıcısı pozitif CNT geçişlerini sayar. 0 = A Zamanlayıcısı •2 vuruşlarını sayar.
6	SPMODE	1 = SERİ PORT çıkışı (CNT kayma saatine kaynaklık eder) 0 = SERİ PORT girişi (dış kayma saatine gerek vardır).
7	TODIN	1 = Doğru zamanlama için TOD ucunda 50Hz'lik saat gerekir. 0 = Doğru zamanlama için TOD ucunda 60Hz'lik saat gerekir.

CRB:

Bit	İsim	İşlevi
		(B Zamanlayıcısı için CRB0-CRB4 bitleri. CRA0-CRA4 bitleri ile aynıdır. Yalnızca 1'inci bit. B ZAMANLAYICISININ PB7 üzerindeki çıkışını kontrol eder
5,6	INMODE	CRB5 ve CRB6 bitleri, B ZAMANLAYICISI için 4 giriş modundan birini seçer:

CRB6	CRB5	
0	0	B zamanlayıcısı •2 vuruşlarını sayar.
0	1	B zamanlayıcısı pozitif CNT geçişleri sayar.
1	0	B zamanlayıcısı, A zamanlayıcısının underflow vuruşlarını sayar.
1	1	B zamanlayıcısı, CNT yüksek iken A zamanlayıcısının underflow vuruşlarını sayar.

Bit	İsim	İşlevi
7	ALARM	1 = TOD kayıtlarına yazıldığında ALARM çalıştırılır. 0 = TOD kayıtlarına yazıldığında TOD saati çalıştırılır.

KAYIT	İSİM	TOD IN	SPMODE	INMODE	LOAD	RUNMODE	OUTMODE	PB ON	START
E	CRA	0=60Hz 1=50Hz	0=INPUT 1=OUTPUT	0= •2 1=CNT	1=FORCE LOAD (STROBE)	0=CONT. 1=O.S.	0=PULSE 1=TOGGLE	0=PB_6 OFF 1=PB_6 ON	0=STOP 1=START

TA (INMODE – START)

KAYIT	İSİM	ALARM	IN	MODE	LOAD	RUNMODE	OUTMODE	PB ON	START
F	CRB	0=TOD 1=ALARM	0 1 1 1	0= •2 1=CNT 0=TA 1=CNT-TA	1=FORCE LOAD (STROBE)	0=CONT. 1=O.S.	0=PULSE 1=TOGGLE	0=PB_7 OFF 1=PB_7 ON	0=STOP 1=START

TB (IN MODE – START)

Kullanılmayan tüm kayıt bitleri yazma işleminden etkilenmez ve okuma sırasında sıfırlanmaya zorlanır.

NOT: COMMODORE SEMICONDUCTOR GROUP, güvenilirliği, işlevi veya tasarımı iyileştirmek amacıyla burada yer alan herhangi bir üründe değişiklik yapma hakkını saklı tutar. COMMODORE SEMICONDUCTOR GROUP, burada açıklanan herhangi bir ürün veya devrenin uygulanması veya kullanımından kaynaklanan hiçbir sorumluluğu kabul etmez; ayrıca kendi patent hakları veya başkalarının hakları kapsamında herhangi bir lisans vermez.

EK N
6566/6567 (VIC-II) ÇİPİ ÖZELLİKLERİ

6566 ve 6567, hem bilgisayar video terminalleri hem de video oyunları için kullanılan çok-amaçlı video renk kontrol cihazlarıdır. Her iki cihazda da 8-bit mikroişlemci veri yolu (65XX) ile erişilebilen 47 kontrol kaydı vardır ve görüntü bilgileri için 16K baytlık bir belleğe erişebilirler. Çeşitli işletim modları ve her bir moddaki seçenekler aşağıda anlatılmaktadır.

KARAKTER GÖSTERİM MODU

Karakter gösterim modunda iken, 6566/6567 belleğin VİDEO MATRİS alanından KARAKTER GÖSTERGEÇLERİNİ alır ve bu göstergeçleri belleğin 2048 baytlık KARAKTER TABANI alanındaki karakter nokta yerleşimi adreslerine aktarır. Video matrisi, her biri 8 bitlik bir karakter göstergeci içeren, art arda 1000 yerleşimden oluşur. Video matrisin bellekteki yeri, video matrisin adresinin en soldaki 4 biti olarak kullanılan, kayıt 24($18) içindeki VM13-VM10 ile tanımlanır. En alt sıradaki 10 bit, 1000 karakter yerleşimini kontrol eden bir iç sayaç (VC3-VC0) ile sağlanır. 6566/6567, 14 adres çıkışı sağlar. Böylece, tüm sistem belleğinin çözümü için ek bir sistem donanımı gerekir.

KARAKTER GÖSTERGECİ ADRESİ

A13	A12	A11	A10	A09	A08	A07	A06	A05	A04	A03	A02	A01	A00
VM13	VM12	VM11	VM10	VC9	VC8	VC7	VC6	VC5	VC4	VC3	VC2	VC1	VC0

Sekiz-bitlik karakter göstergeci, 256 değişik karakter tanımının aynı anda kullanılabilmesini sağlar. Her karakter, karakter tabanında, art arda gelen sekiz baytta saklanan, 8x8'lik bir matristir. Karakter tabanının yerleşimi, kayıt 24($18) içinde yer alan ve karakter taban adresinin en sol üç biti olarak kullanılan CB13-CB11 ile tanımlanır. En alt 11 adres, video matrisinde, belirli bir karakteri seçen 8-bitlik, karakter göstergeci (D7-D0) ve sekiz karakter baytından birini seçen 3-bitlik raster sayacı (RC2-RC1) ile oluşturulur. Elde edilen karakterler, her biri 40 karakter içeren 25 satırmış gibi biçimlendirilir. 8-bitlik karakter göstergecine ek olarak, her video matris yerleşimi (video matris belleği 12 bit genişliğindedir) ile her karakter için onaltı renkten birini tanımlayan 4-bitlik RENK YARI-BAYTI vardır.

KARAKTER VERİ ADRESİ

A13	A12	A11	A10	A09	A08	A07	A06	A05	A04	A03	A02	A01	A00
CB13	CB12	CB11	D7	D6	D5	D4	D3	D2	D1	D0	RC2	RC1	RC0

STANDART KARAKTER MODU (MCM = BMM = ECM = 0)

Standart karakter modunda, karakter tabanının art arda gelen 8 baytı, direkt olarak her karakter alanının 8 satırında görüntülenir. Bir bitin "0" olması, 0 nolu zemin renginin (kayıt 33($21)'den) görüntülenmesine neden olur. Bitin "1" olması ise renk yarı-baytı (ön-plan) ile seçilen rengin görüntülenmesini sağlar.

FONKSİYON	KARAKTER BİTİ	GÖRÜNTÜLENEN RENK
Zemin	0	0 nolu zemin rengi (kayıt 33($21)
Ön-Plan	1	4-bitlik renk yarı-baytı ile seçilen renk

Böylece, her karakterin 4-bitlik renk yarı-baytı (16 renkten biri) tanımlanan tek bir rengi vardır ve tüm karakterler aynı zemin rengini paylaşırlar.

ÇOK-RENKLİ KARAKTER MODU (MCM = 1, BMM = ECM = 0)

Çok renkli mod, her karakter için 4 renge kadar, ek renk esnekliği sağlar, fakat çözünürlük azalır. Çok-renkli mod, kayıt 22($16)'daki MCM biti "1" yapılarak seçilir. Bu, karakter tabanında saklanan nokta şeklindeki verilerin değişik bir biçimde yorumlanmasına neden olur. Renk yarı-baytının en soldaki biti "0" ise, karakter, standart karakter modunda tanımlandığı şekilde görüntülenecek ve iki modun birleştirilmesine izin verecektir (fakat yalnızca, en alt 8 renk kullanılabilir). Renk yarı-baytının en soldaki biti "1" olduğunda (eğer MCM:MSB(CM) = 1 ise), karakter bitleri çok-renkli modda ve aşağıdaki gibi yorumlanacaklardır:

FONKSİYON	KARAKTER BİTİ ÇİFTİ	GÖRÜNTÜLENEN RENK
Zemin	00	0 nolu zemin rengi (kayıt 33($21)
Zemin	01	1 nolu zemin rengi (kayıt 34($22)
Ön-Plan	10	2 nolu zemin rengi (kayıt 35($23)
Ön-Plan	11	Renk yan-baytının en sağdaki 3 biti ile belirlenen renk.

Bir nokta rengini belirlemek için iki bit gerektiğinden, her karakter 4 x 5'lik matrisler halinde görüntülenir. Çünkü her noktanın yatay boyutu, standart moddakinin iki katıdır. Fakat bu durumda her karakter bölgesi 4 renk içerir, iki tane zemin için, iki tane de ön-plan için (Yaratık önceliklerine bakınız).

GELİŞTİRİLMİŞ RENK MODU (ECM = 1, BMM = MCM = 0)

Bu renk modu, normal 8 x 8 karakter çözünürlüğündeki her bir karakter bloğunun zemin renginin, teker teker tanımlanabilmesine olanak verir. Bu mod, kaydı 17($11)'deki ECM biti "1" yapılarak seçilir. Karakter nokta verisi standart moddaki gibi görüntülenir (renk yarı-baytı ile tanımlanan ön-plan rengi, veri biti "1" olduğunda görüntülenir), fakat karakter göstergecinin en sol 2 biti, her karakter bölgesinin zemin rengini aşağıda gösterildiği şekilde seçmek için kullanılır:

KARAKTER GÖSTERGECİNİN EN SOLDAKİ BİT ÇİFTİ	BİT 0 İÇİN GÖRÜNTÜLENEN ZEMİN RENGİ
00	0 nolu zemin rengi (kayıt 33($21)
01	1 nolu zemin rengi (kayıt 34($22)
10	2 nolu zemin rengi (kayıt 35($23)
11	3 nolu zemin rengi (kayıt 36($24)

Renk bilgileri için, karakter göstergecinin en sol iki biti kullanıldığından, yalnızca 64 değişik karakter tanımı kullanılabilir. 6566/6567 içerdikleri orijinal göstergeç değerlerinin ne olduğunu önemsemeden, CB10, CB9'u "0" olması için zorlar. Bu yüzden yalnızca ilk 64 karakter tanımına erişilebilir. Genişletilmiş renk modunda her karakter, teker teker tanımlanmış olan 16 zemin renginden birine sahip olabilir.

NOT: Geliştirilmiş renk modu ve çok-renkli mod ile aynı anda çalışmak mümkün değildir

BİT HARİTA MOD'U

Bit harita modunda, 6566/6567 verilen bellekten daha değişik bir tarzda alır. Bu durumda, görüntülenen nokta ile bellek biti arasında bire-bir eşleşme söz konusudur. Bit harita modu, her bir noktanın görüntüsünün kontrol edilebildiği 320 yatay X 200 dikey ekran çözünürlüğü sağlar. Bit harita Modu kaydı 17($11) içindeki BMM biti “1” yapılarak seçilir. VİDEO MATRİS'ine karakter modunda olduğu gibi bu modda da erişmek mümkündür, fakat, video matris verileri, karakter göstergeçleri gibi uzun süre yorumlanmazlar. VİDEO MATRİS SAYACI, noktanın verisini alıp, 8000 GÖRÜNTÜ TABANINDA görüntülemek için, bir adres olarak da kullanılabilir. Görüntü taban adresleri aşağıdaki şekilde oluşturulur.

A13	A12	A11	A10	A09	A08	A07	A06	A05	A04	A03	A02	A01	A00
CB13	VC9	VC8	VC7	VC6	VC5	VC4	VC3	VC2	VC1	VC0	RC2	RC1	RC0

VC_X: video matris sayacının çıktıları, RC_X: 3-bitlik raster satır sayacı anlamında ve CB13’de: kayıt 24($18)'den anlamına gelir. Video matris sayacı, sekiz-raster satırı için aynı 40 yerleşim üzerinde adım adım ilerler ve her sekiz satır için bir sonra gelen 40 yerleşim üzerinden devam eder. Bu arada raster sayacı, her bir yatay video satırı (raster satırı) için, birer birer arttırılmaktadır. Bu adresleme, art arda gelen her bir 8 bellek yerleşiminin, video görüntüsü üzerinde 8 X 8"1ik nokta blokları şeklinde biçimlenmesiyle sona erer.

STANDART BİT HARİTA MOD'U (BMM = 1, MCM = 0)

Standart bit harita madunu kullanıyorsanız, renk ile ilgili bilgiler yalnızca, video matriste bulunan verilerden elde edilebilir (renk yarı-baytı (nybble) hesaba katılmaz). 8 bit, iki,1-bitlik yarı-bayta bölünür. Bu şekilde, her bir 8 x 8’lik nokta bloğu için, birbirinden bağımsız iki renk seçilebilir. Görüntü belleğindeki bir bit “0” olduğunda, noktanın rengi, en sağdaki (alt) yarı-bayt (LSN: Least significant nybble) ile tanımlanır. Aynı şekilde, o görüntü belleğindeki bitin “1” olması en-soldaki (üst) yan-bayt (üst nybble) (MSN: Most significant nybble) ile belirlenen rengin seçilmesini sağlar.

BİT	GÖRÜNTÜ RENGİ
0	Video matris göstergecinin en-alt yarı-baytı
1	Video matris göstergecinin en-üst yarı-baytı

ÇOK-RENKLİ BİT HARİTA MODU (BMM = MCM = 1)

Çok-renkli bit harita modu, BMM biti ile birlikte, kayıt 22($16) içindeki MCM bitinin “1” yapılmasıyla seçilir. Çok-renkli mod, standart bit harita moduyla aynı bellek erişim sırasını kullanır, fakat nokta verilerini aşağıdaki gibi yorumlar:

BİT	GÖRÜNTÜ RENGİ
00	0 nolu zemin rengi (Kayıt 33($21)
01	Video matris göstergecinin en-üst yarı-baytı
10	Video matris göstergecinin en-alt yarı-baytı
11	Video matrisin renk yarı-baytı

Renk yarı-bayt, (DB11-DB8) IS'nin, çok-renkli bit harita modu için kullanıldığına dikkat edin. Yine, bir nokta rengini seçmek için iki bit kullanılır, yatay noktanın boyutu çifttir ve bu ekran çözünürlüğünün 160 yatay x 200 dikey olmasına neden olur. Çok-renkli bit harita modunun kullanılması, her bir 8 x 8'1ik bloğun, zemin rengine ek olarak, birbirinden bağımsız olarak seçilen 3 ayrı renkte görüntülenebilmesini sağlar.

HAREKET EDEBİLEN NESNE BLOKLARI (MOB, Yaratıklar)

Bundan sonraki bölümlerde MOB olarak kısalttığımız, MOVABLE OBJECT BLOCKS, yan, Hareket edebilir Nesne Blokları: ekranın herhangi bir konumunda görüntülenebilen özel bir karakter türüdür. Bit harita modu ve karakterler için mevcut olan blok kısıtlamaları, MOB'lar için söz konusu değildir. 8 tane MOB un aynı anda, görüntülenmesi mümkündür. MOB'ların her biri bellekte 63 baytla tanımlanır ve 24 x 21 noktalık matrisler halinde (aşağıda gösterildiği gibi) görüntülenirler. Birçok özel kolaylıklar nedeniyle MOB'lar video grafikleri ve oyunlar için oldukça kullanışlıdır.

MOB GÖRÜNTÜ BLOĞU

BAYT	BAYT	BAYT
00	01	02
03	04	05
--	--	--
--	--	--
--	--	--
57	58	59
60	61	62

ETKİN DURUMA GETİRME

Her bir MOB'un görüntülenmesi için, kayıt 21($15)'de o MOB'a karşılık gelen etkin biti (MnE) "1" yapmak gerekir. Eğer MnE biti 0 ise, pasif edilen MOB için hiçbir MOB işlemi yapılmaz.

KONUM

Her MOB, kendi X ve Y konum kayıtları (kayıt haritasına bakın) aracılığıyla 256 dikey çözünürlükte yerleştirilebilir. MOB'un yeri, matrisin sol-üst köşesi ile belirtilir. 23'ten 347'ye ($17-$157) kadar olan X yerleşimleri ve 50'den 249'a ($32-$F9) kadar olan Y yerleşimleri görülebilir niteliktedir. Kullanılabilen bütün MOB konumları, ekran üzerinde tümüyle görülebilir olmadığından, MOB'ta, görüntü ekranına ya da ekranın dışına doğru yavaşça hareket ettirilebilir.

RENK

MOB rengini belirlemekte her MOB için ayrı ayrı 4-bitlik kayıt kullanılır. MOB renk modları:

STANDART MOB (MnMC = 0)

Standart modda, MOB verisinde bir bitin "0" olması rengin saydam olmasını yani geçirgen olmasını sağlar. Bunun anlamı zemindeki veriyi geçirmesidir.

"1" olan bit, ise karşılık gelen MOB renk kaydı ile tanımlanan MOB rengi ile gösterilir.

ÇOK-RENKLİ MOB (MnMC = 1)

Her MOB ayrı ayrı, MOB Çok-Renkli kayıtlarındaki (kayıt 28($1C)), MnMC bitleri aracılığıyla, çok-renkli MOB olarak seçilebilir. MnMC biti “1” ise, bu bite karşılık gelen MOB çok-renkli modda görüntülenir. Çok-renkli modda, MOB verisi çifter çifter yorumlanır (diğer çok-renkli modlardaki gibi).

BİT ÇİFTİ	GÖRÜNTÜLENEN RENK
00	Saydam
01	MOB çok-renkli No 0 (kayıt 37($25))
10	MOB rengi (39-46($27-$2E)) arasındaki kayıtlar
11	MOB çok-renkli No 1 (kayıt 38($26))

Her renk için iki bit gerektiğinden, MOB’un çözünürlüğü 12 x 21’e düşer. Her yatay noktanın büyüklüğü 2 katına çıktığı için ise, MOB’un büyüklüğü değişmez. Her MOB için 3 renk kullanılabilir (saydam renge ek olarak). Fakat bu renklerden 2’si, çok-renkli moddaki tüm MOB’lar tarafından paylaşılır.

BÜYÜKLÜK

Her bir MOB ayrı ayrı hem yatay hem de dikey olarak 2 misli genişletilebilir. Büyüklük kontrolü için, iki kayıttaki kontrol bitleri (MnXE, MnYE) kullanılır:

KAYIT	İŞLEVİ
23 ($17)	Yatay genişleme MnXE – “1” = geniş; “0” = normal
29 ($1D)	Dikey genişleme MnYE – “1” = geniş; “0” = normal

MOB’lar genişletildiğinde, çözünürlükte herhangi bir artış göze çarpmaz. Aynı 24 x 21’lik (çok-renkli modda 12 x 21) matris görüntülenir, fakat tüm MOB boyutu istenilen yönde iki katına çıkartılır (eğer MOB hem çok-renkli hem de genişletilmiş ise, en küçük MOB noktası, standart nokta boyutunun 4 katı büyüklüğünde olacaktır).

ÖNCELİK

Her MOB'un önceliği, bit harita madunun ya da karakterin diğer görüntüleme bilgilerine bağlı olarak, ayrı ayrı kontrol edilir. Her bir MOB'un önceliği, kayıt 27($1B) de, bu MOB'a karşılık gelen MnOP biti aşağıda gösterildiği gibi set edilerek belirlenir:

KAYIT BİTİ	KARAKTER YA DA BİT HARİTA VERİSİ ÖNCELİĞİ
0	Saydam olmayan MOB verisi görüntülenecek (MOB ön planda)
1	Saydam olmayan MOB verisi, yalnızca Zemin No 0 ya da Çok-renkli bit çitti 01 'in yerine görüntülenecektir (MOB arkada)

MOB GÖRÜNTÜLENEN VERİ ÖNCELİĞİ

MnDP = 1	MnDP = 0
MOBn Ön plan Zemin	Ön plan MOBn Zemin

“0” olan MOB veri bitleri (çok-renkli modda 00”) saydamdırlar ve diğer herhangi bir bilginin görüntülenmesine izin verirler.

Her MOB'un diğerine göre sabit bir önceliği vardır; MOB 0’ın en yüksek önceliğe, MOB 7’nin en düşük önceliğe sahip olması gibi. İki MOB’un verileri çakıştığında (saydam veriler hariç), en düşük numaralı MOB’un verileri görüntülenecektir.

ÇARPIŞMANIN ALGILANMASI

MOB'lar iki şekilde çakışabilirler ya da çarpışabilirler: MOB ile MOB çarpışması ve MOB ile veri çarpışması:

1) İki MOB arasındaki çarpışma, MOB'ların (non-transparent) çıkış verileri karşılaştığı zaman oluşur. MOB (transparent) alanları çakıştığında çarpışma meydana gelmez. Karşılaşma olduğunda, MOB-MOBCOLLISION (çarpışma) kaydı 30($1E)'deki MOB bitleri (MnM), iki MOB'unda çarpıştığını belirtmek için "1" olur. Bir veya birden fazla MOB çarpıştığında, her MOB için bir bit "1" olur. Bu bitler, kayıtların, otomatik olarak sıfırlanması amacıyla okunmasına kadar bir olarak kalır. MOB çarpışmaları ekran dışında olsa bile sezilebilir.

2) İkinci tip çarpışma, MOB ve ön plan görüntü verisi arasında oluşur. MOB-VERİ ÇARPIŞMA kaydı 31($1F)'de, her ön plan görüntü verisi ile karşılaşabilecek MOB için "1" olan (MnD) bitleri bulunur. Yukarıda bahsedildiği gibi gene transparent verilerin karşılaşması, çarpışma meydana getirmez. Özel uygulamalar için, 0-1 çok renkli bit çiftinden gelen görüntü verisinde çarpışmaya sebep olmaz. Bu özellik, gerçek MOB çarpışmaları ile karıştırmadan, ton görüntü verisi olarak kullanılmalarını sağlar. MOB-VERİ çarpışması, gerçek görüntü verisi ekran dışı pozisyona kaydığı zaman, yatay olarak ekran dışında oluşabilir. MOB-VERİ ÇARPIŞMA kaydı da otomatik olarak okunarak sıfırlanır.

Çarpışma önleme bitleri, iki kayıttan birisinin ilk biti "1" olduğunda 1 durumuna geçer. Kayıttaki herhangi bir çarpışma biti "1" olduğunda, sonraki çarpışmalar kayıt okuma ile tüm bitleri "0" olana kadar, önleme bitini yüksek seviyeye çıkarmayacaktır.

MOB BELLEK ERİŞİMİ

Her MOB'un verisi 63 ardışık bellek baytına saklanır. MOB veri blogu, VİDEO MATRİSİ'nin sonunda bulunan MOB göstergeci ile tanımlanır. Video matrisinin sadece 1000 baytı, 1016-1025 matris yerlerinin MOB göstergeçlerine verilmesiyle normal görüntü modunda kullanılır. Matrisdeki 8 bitlik MOB göstergeciyle, 6 bit MOB bayt sayacı (63 baytı adreslemek için), 14 bit adres alanının tümünü tanımlar:

A13	A12	A11	A10	A09	A08	A07	A06	A05	A04	A03	A02	A01	A00
MP7	MP6	MP5	MP4	MP3	MP2	MP1	MP0	MC5	MC4	MC3	MC2	MC1	MC0

Burada MPx, video matrisinden gelen MOB göstergeç bitleri, MCx ise MOB sayaç bitleridir. MOB göstergeçleri her raster satırı sonunda, video matrisinden okunur. MOB'un Y konum kaydı, o anki raster satır sayımı ile aynı değeri aldığında, MOB verisinin gerçek fetch'i başlar. İç sayaçlar, her raster satırında üç bayt göstererek, 63 baytlık MOB verisine ulaşır.

DİĞER ÖZELLİKLER

EKRANIN BOŞALTILMASI

Görüntü ekranı, kayıt 17($11)'deki DEN biti "0" yapılarak boşaltılabilir. Ekran boşaltıldığında tüm ekran, kayıt 32($20)'deki renk ile kaplanacaktır. Ekranın boşaltılmasına çalışılırken, sadece görünmeyen (Faz 1) bellek erişimi gereklidir ve sistem veri yolunun işlemcinin tümünü kullanmasına izin verilir. MOB verilerine, MOB'lar pasif edilmemişse erişilebilir. Normal video gösterimi için DEN bitinin "1" yapılması gerekir.

SATIR/KOLON SEÇİMİ

Normal görüntü, 25 satır ve her satır satırdaki 40'ar karakterden oluşur. Özel gösterimler için, görüntü penceresi 24 satır ve her satırda 38 karaktere indirgenebilir. Görüntü bilgilerinin formatında herhangi bir değişiklik olmaz. Bu durumda yalnızca, dış çerçeve alanına yapışık olan karakterler (bitler), çerçeve ile örtülecektir. Bitlerin seçilmeleri aşağıdaki gibidir:

RSEL	SATIR SAYISI	CSEL	KOLON SAYISI
0	24 Satır	0	38 Kolon
1	25 Satır	1	40 Kolon

RSEL biti kayıt 17($11)'de, CSEL biti ise kayıt 22($16)'dadır. Normalde standart görüntü için daha büyük olan pencere kullanılır. Daha küçük olan pencere ise, normalde, kaydırma ile beraber kullanılır.

KAYDIRMA

Görüntülenen veri, ekran üzerinde yukarıdan-aşağıya ve sağdan-sola, tüm bir karakter boşluğu boyunca kaydırılabilir. Daha küçük olan görüntü penceresi (yukarıda bahsetmiştik) ile beraber kullanıldığında, sistem belleği yeniden düzenlenirken (yalnızca yeni bir karakter satırı ya da kolonu gerektiğinde), görüntü verisinin yumuşak olarak hareket etmesini sağlamak amacıyla kaydırma kullanılabilir. Kaydırma aynı zamanda görüntü penceresi üzerinde sabit bir görüntüyü ortalamak için de kullanılabilir:

BİTLER	KAYIT	FONKSİYON
X2, X1, X0	22($16)	Yatay Konum
Y2, Y1, Y0	17($11)	Dikey Konum

LIGHT-PEN (Işık Kalemi)

Lıght-pen girişi, o anda ekranda bulunan konumu, bir çift kayda düşen-kenarda (lov-going edge) kaydeder. X konumu kaydı (kayıt 19($13)), geçiş anındaki X konumunun en soldaki 8 bitini içerecektir. X konumu 512-durum sayacı ile tanımlandığından, 2 yatay nokta hassasiyeti sağlanmıştır. Aynı şekilde, Y konumu da kayıt 20($14)'e kaydedilir. Fakat burada 8 bit, görülebilen görüntü üzerinde tek raster hassasiyeti sağlarlar. Light-pen mandalı her çerçevede bir kez tetiklenebilir ve aynı çerçeve üzerindeki diğer tetiklemelerin herhangi bir önemi yoktur. Böylece, light-pen'i ekran üzerinde döndürerek, light-pen'inizin yapısına bağlı olarak birçok örnek alabilirsiniz (ortalama, 3 ya da daha fazla örnek).

RASTER KAYDI:

Raster kaydı çift-fonksiyonlu kayıttır. Raster kaydı 18($12) okunduğunda elde edilen değer. O andaki raster konumunun (Kayıt 17($11)'da yer alan RC8 en soldaki bit) en alt 8 bitidir. Raster kaydı, görüntünün titremesini önlemek amacıyla, görülebilir alan dışındaki görüntü değişikliklerini yerine getirmek için kullanılabilir. Görünebilir ekran penceresi raster 51 ile raster 251 ($033-$0FB) arasındadır. Raster bitlerine (RC8 dahil) uygulanacak bir yazma işlemi, iç raster kıyaslamalarında kullanılmak üzere kaydedilir. O andaki raster ile yazılmış olan değer birbirine uygun ise raster kesinti latch'i set edilir (yani bir olur).

KESİNTİ KAYDI

Kesinti kaydı, kesintinin dört kaynağının durumunu gösterir. Kesinti kaynağı, kesinti için bir talepte bulunduğunda, kayıt 25($19) içindeki kesinti latch biti “1” olur. Dört kesinti kaynağı:

LATCH BİTİ	AKTİF BİTİ	SET EDİLDİĞİNDE
IRST	ERST	(raster sayısı) = (saklanmış raster sayısı) olduğunda set edilir.
IMDC	EMDC	MOB-VERİ çarpışma kaydı ile set edilir. (Yalnızca ilk çarpışma için)
IMMC	EMMC	MOB-MOB çarpışma kaydı ile set edilir. (Yalnızca ilk çarpışma için)
ILP	ELP	Light-pen girişinin negatif geçişi için set edilir. (Her çerçeve için bir kere)
IRQ		Latch set edildiğinde ve aktif edildiğinde yüksek olarak set edilir. (IRQ-Çıkış’ın tersi)

IRQ'yu set etmek, çıktıyı “0”’lamak ya da bir kesinti isteğini aktif etmek için karşılık gelen kesinti aktif bitinin (kayıt 26($14) içinde yer alan) “1” yapılması gerekir. Bir kere bir kesinti latch’i set edildiğinde, latch yalnızca, kesinti kaydı içinde yer alan ve istenilen latch’e “1” yazılarak temizlenebilir. Bu özellik, yazılımın aktif kesintileri “hatırlamasına” gerek duyulmaksızın video kesintilerinin seçilerek değerlendirmesini sağlar.

DİNAMİK RAM'IN TAZELENMESİ (DYNAMIC RAM REFRESH)

Dinamik RAM tazeleme kontrolörü, 6566/6567 cihazlarının içine yerleştirilmiştir. Beş 8-bitlik satır adresi, her raster satırında tazelenir. Bu oran, 128 tazeleme şeması içinde herhangi bir tek satır adresleri arasında en çok 2.02ms’lik bir gecikmeyi garanti eder. (256 adreslik tazeleme şemasında, en fazla gecikme 3.66ms'dir.) Bu tazeleme sistem için tümüyle saydamdır, çünkü, tazeleme sistem saatinin 1'inci Fazı süresince oluşur. 6567, normal olarak dinamik RAM’lere doğrudan bağlanan RAS/ ve CAS/'leri üretir. RAS/ ve CAS/ her 2’nci Faz ve her video verisine erişim (tazelemeyi içerir) için üretilir. Böylece dış saat üretimi gerekmez.

ÇALIŞMA TEORİSİ

SİSTEM ARABİRİMİ

6566/6567 video kontrol cihazları, sistem veri yolu ile özel bir yoldan ilişkide bulunurlar. Bir 65XX sistemi, yalnızca çevrimin Faz 2 (saat yüksek düzeyde) sırasında, sistem veri yoluna gerek duyar. 6566/6567 cihazları, saat çevriminin Faz 1 (saat alçak düzeyde) sırasında da normal olarak sistem belleğine erişebilme avantajına sahiptir. Bu yüzden, bellek tazeleme ve karakter verilerini alıp getirme gibi işlemler, işlemci için oldukça açıktır ve bu işlemler işlemcinin verimini azaltmazlar. Video çipleri, bu veri yolu paylaşımını elde etmek için gerekli olan arabirim kontrol sinyallerini sağlarlar.

Video cihazları, AEC (Address Enable Control: adresin geçerliliği kontrolü) sinyalini sağlarlar ve bu sinyal, video cihazının adres veri yoluna erişmesine izin vererek işlemci adres veri yolu sürücülerini pasif etmekte kullanılır. AEC, 65XX ailesi AEC girişlerine doğrudan bağlanmasını sağlayacak şekilde aktif alçak düzeydedir.

AEC sinyali normal olarak. Faz 1 sırasında harekete geçirilir ve böylece, işlemcinin çalışması bundan etkilenmez. Bu veri yolu "paylaşımı" nedeniyle, tüm bellek erişimleri 1/2 çevrim içerisinde tamamlanmak zorundadır. Video çipleri 1 MHz'lik saat (bu, sistem Faz 2 olarak kullanılmalıdır) sağladıklarından, bir bellek çevrimi, adres yerleştirmeleri, veri erişimleri ve okunan cihaza veri yerleştirmelerini de içerecek şekilde, 500ns'dir.

6566-6567'nin belirli bazı işlemleri, faz 1 süresi içinde okuma ile mümkün olandan daha hızlı veri akışına gerek duyabilir: özellikle video matrisinden karakter göstergeçlerine erişmek ve MOB verisini almak gibi işlemler için bu durum söz konusudur. Buna göre, işlemcinin çalışmasının durdurulması ve veri erişiminin faz 2 saati süresince yapılması gerekir. Bu da ancak BA (veri yolu kullanılabilir) sinyali ile sağlanabilir. BA hattı, normalde yüksek düzeydedir. Fakat, video çipinin faz 2 veri erişimini gereksindiğini göstermek amacıyla, faz 1 süresince alçak düzeye getirilir. BA'nın alçak düzeye getirilmesinden sonra, işlemcinin o anki bellek erişimini tamamlaması için, üç kez faz 2 zamanına izin verilir. BA alçak düzeye geldikten sonra, dördüncü faz 2 zamanı AEC sinyali, video çipi veri alacağı için alçak düzeyde kalır. BA hattı normalde 65XX işlemcisinin RDY girişine bağlıdır. Karakter göstergecinin alınması için, 40 ardışık faz 2 erişimine gerek duyulur. MOB verisinin elde edilmesi için 4 türlü bellek erişimine gerek vardır.

FAZ	VERİ	DURUM
1	MOB göstergeci	Her rasterde
2	MOB bayt 1	MOB görüntülenirken her bir rasterde
3	MOB bayt 2	MOB görüntülenirken her bir rasterde
4	MOB bayt 3	MOB görüntülenirken her bir rasterde

Her raster satırının sonunda MOB göstergeçleri, diğer Faz 1 sürelerinde alınır. Bu amaçla ek çevrimlere gerek duyulur. Her zaman olduğu gibi gerekli tüm veri yolu kontrolü 6566/6567 cihazları tarafından sağlanır.

BELLEK ARABİRİMİ

Video arabirim çipinin iki şekli, 6566 ve 6567, adres çıkışı konfigürasyonlarında birbirinden farklıdırlar. 6566'da sistem adres veri yoluna doğrudan bağlanabilmesi için tümü çözümlü onüç adres vardır. 6567'de ise 64K dinamik RAM'e doğrudan bağlanabilmesi için, çok düzeyli (multiplexed) adresler vardır. En sağdaki adres bitleri A06-A00, RAS/alçak seviyedeyken A06-A00 üzerinde, en soldaki bitler A13-A08 ise CAS alçak seviyedeyken A05-A00 üzerindedir. 6567 üzerindeki A11-A00 uçları, bu bitlerin konvensiyonel 16K(2K x 8) ROM'a doğrudan bağlanmalarını sağlayan statik adres çıkışlarıdır. (Alt basamak adreslerinin dış latching (kilitlenme) gereksinimi vardır)

İŞLEMCİ ARABİRİMİ

Yukarıda bellek erişimlerinin özellikleri anlatılan 6566/6567 kayıtları herhangi bir çevre-birimi cihazına aynı şekilde erişirler. Aşağıda işlemci arabirim sinyalleri verilmiştir:

VERİ YOLU (DB7-DB0)

Veri yolunun sekiz ucu, çift-yönlü bir veri portudur ve CS/, RW, ve Faz 0 ile kontrol edilir. Veri yoluna yalnızca AEC ve Faz 0 yüksek, CS/ alçak iken erişilebilir.

ÇİP SEÇİMİ (CSI)

Çip seçme ucu CS/, adres ve RW uçları ile beraber cihaz kayıtlarına erişimi mümkün kılmak amacıyla alçak düzeye getirilir. CS/ alçak yalnızca, AEC ve Faz 0 yüksek iken tanımlanabilir.

OKUMA/YAZMA (R/W)

Okuma/yazma girişi R/W, CS/ ile beraber, veri yolu üzerindeki veri aktarımının yönünü belirtmek için kullanılır. R/W yüksek ("1") olduğu zaman veri, seçilmiş olan kayıttan, veri yolu çıkışına aktarılır. R/W düşük ("0") olduğunda ise veri yolu uçları üzerinde yer alan veri, seçilmiş olan kayda yüklenir.

ADRES VERİ YOLU (A05-A00)

En alt alt, adres ucu (A5·A0), çift-yönlüdür. Video cihazının, işlemci okuma ya da yazma sırasında, bu adres uçları giriş olarak kullanılır. Adres girişleri üzerindeki veri kaydı, kayıt haritasında tanımlandığı gibi okuma ya da yarma için seçer.

SAAT ÇIKIŞI (PH0)

Saat çıkışı, Faz 0, 65XX işlemcisi Faz 0 girişi için kullanılan 1MHz frekanslı saattir. Tüm sistem veri yolu aktiviteleri, bu saate bağlı olarak çalışır. Saat frekansı 8 MHz'lik video giriş saatinin 8'e bölünmesiyle elde edilir.

KESİNTİLER (IRQ/)

Kesinti çıkışı IRQ/, cihaz içinde etkin bir kesinti kaynağı oluştuğunda düşük düzeye getirilir. IRQ/ çıkışı açık drenajdır ve harici bir çekme direnci gerektirir.

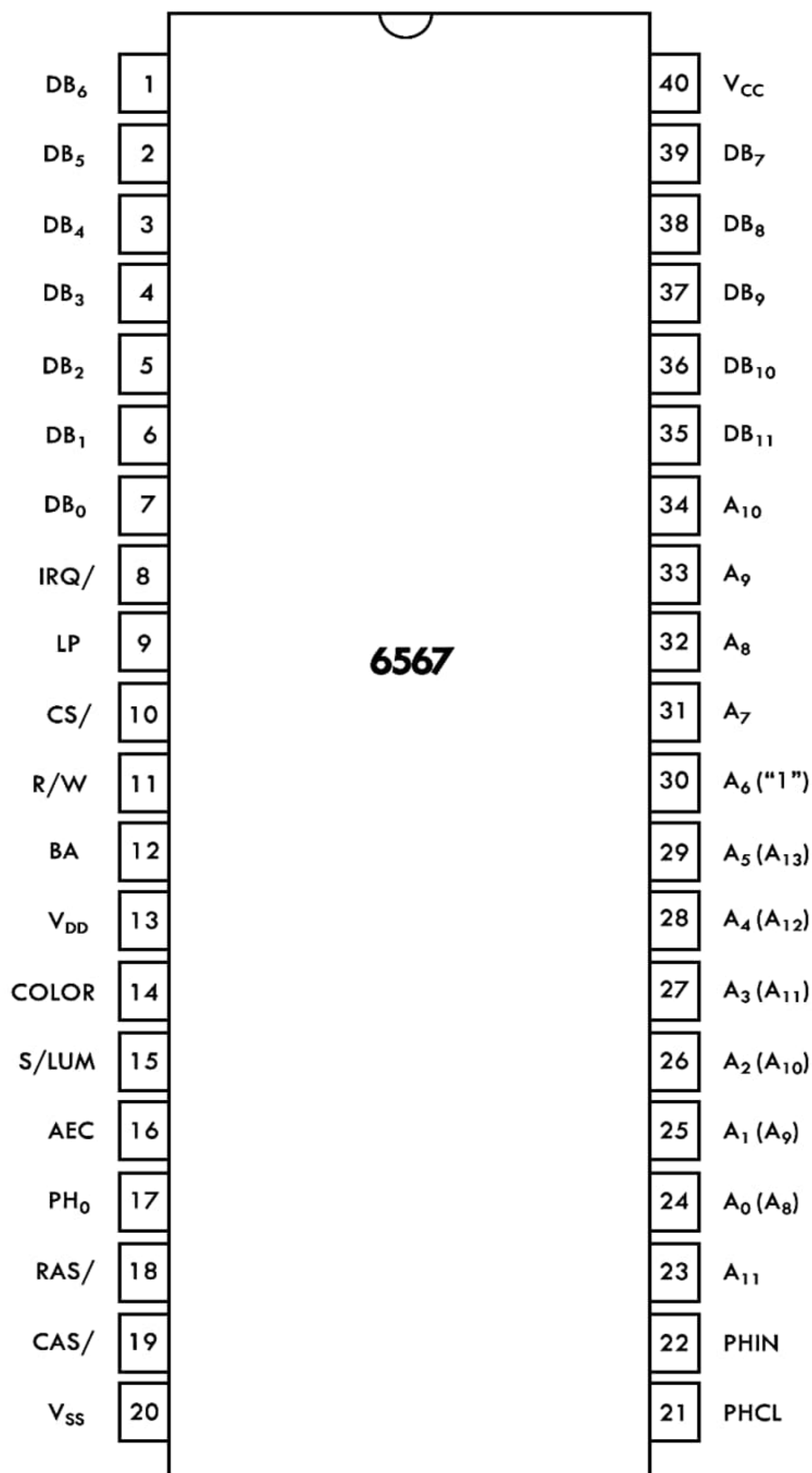
VİDEO ARABİRİMİ

6566 ve 6567'deki video çıkış sinyali, dışarıda birbirleriyle karıştırılmaları gereken iki sinyalden oluşmuştur. SYNC/LUM çıkışı, yatay ve dikey senkronizasyon sinyalleri, video görüntüsünün luminance (parlaklık) bilgileri gibi tüm video verilerini içerir SYNC/LUM, 500ohm'luk bir dış çekme direnci gerektiren açık yoldur. COLOR (renk) çıkışı, görüntülenen verilerin tümünün rengi gibi tüm chrominance bilgilerini içerir. COLOR çıkışı açık kaynaktır ve toprakla arasına 1000 ohm'luk bir direnç yerleştirilmesi gerekir. Bu iki sinyalin uygun olarak karıştırılmasıyla oluşan sinyal, bir video monitörüne doğrudan bağlanabilir ya da normal bir televizyon ile birlikte kullanmak için bir modülatöre bağlanabilir.

6566/6567 VERİ YOLU AKTİVİTELERİ ÖZETİ

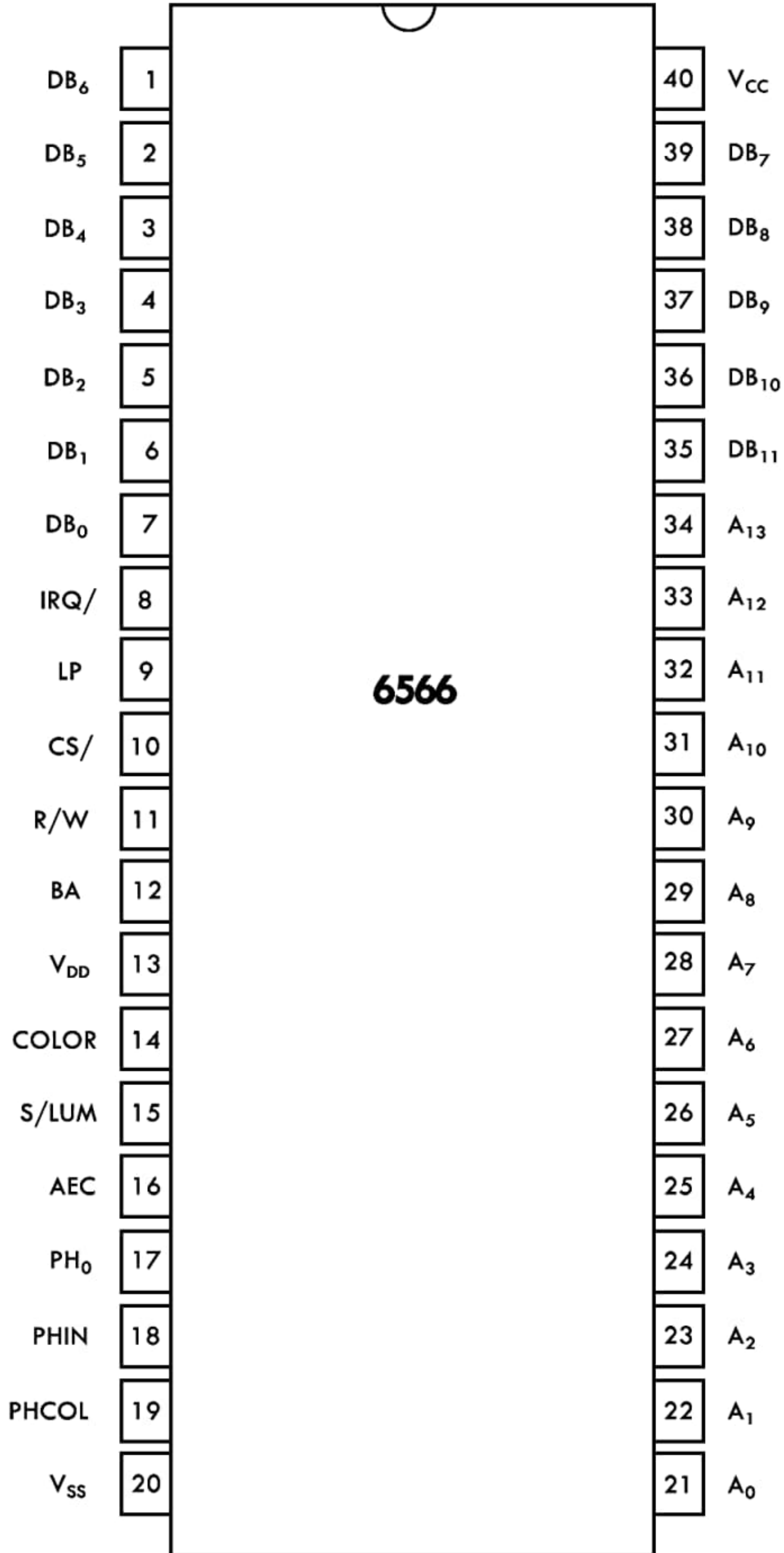
AEC	PH0	CS/	R/W	HAREKET
0	0	X	X	FAZ 1 gidip getrime, tazeleme
0	1	X	X	FAZ 1 gidip getrime (işlemci kapalı)
1	0	X	X	Hareket yok
1	1	0	0	Seçilen kayda yaz
1	1	0	1	Seçilen kayıttan oku
1	1	1	X	Hareket yok

BACAK DİZİMİ



Çoklu adresler parantez içinde verilmiştir.

BACAK DİZİMİ



KAYIT HARİTASI

ADRES		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	TANIMI
00	($00)	M0X7	M0X6	M0X5	M0X4	M0X3	M0X2	M0X1	M0X0	MOB 0 X-konumu
01	($01)	M0Y7	M0Y6	M0Y5	M0Y4	M0Y3	M0Y2	M0Y1	M0Y0	MOB 0 Y-konumu
02	($02)	M1X7	M1X6	M1X5	M1X4	M1X3	M1X2	M1X1	M1X0	MOB 1 X-konumu
03	($03)	M1Y7	M1Y6	M1Y5	M1Y4	M1Y3	M1Y2	M1Y1	M1Y0	MOB 1 Y-konumu
04	($04)	M2X7	M2X6	M2X5	M2X4	M2X3	M2X2	M2X1	M2X0	MOB 2 X-konumu
05	($05)	M2Y7	M2Y6	M2Y5	M2Y4	M2Y3	M2Y2	M2Y1	M2Y0	MOB 2 Y-konumu
06	($06)	M3X7	M3X6	M3X5	M3X4	M3X3	M3X2	M3X1	M3X0	MOB 3 X-konumu
07	($07)	M3Y7	M3Y6	M3Y5	M3Y4	M3Y3	M3Y2	M3Y1	M3Y0	MOB 3 Y-konumu
08	($08)	M4X7	M4X6	M4X5	M4X4	M4X3	M4X2	M4X1	M4X0	MOB 4 X-konumu
09	($09)	M4Y7	M4Y6	M4Y5	M4Y4	M4Y3	M4Y2	M4Y1	M4Y0	MOB 4 Y-konumu
10	($0A)	M5X7	M5X6	M5X5	M5X4	M5X3	M5X2	M5X1	M5X0	MOB 5 X-konumu
11	($0B)	M5Y7	M5Y6	M5Y5	M5Y4	M5Y3	M5Y2	M5Y1	M5Y0	MOB 5 Y-konumu
12	($0C)	M6X7	M6X6	M6X5	M6X4	M6X3	M6X2	M6X1	M6X0	MOB 6 X-konumu
13	($0D)	M6Y7	M6Y6	M6Y5	M6Y4	M6Y3	M6Y2	M6Y1	M6Y0	MOB 6 Y-konumu
14	($0E)	M7X7	M7X6	M7X5	M7X4	M7X3	M7X2	M7X1	M7X0	MOB 7 X-konumu
15	($0F)	M7Y7	M7Y6	M7Y5	M7Y4	M7Y3	M7Y2	M7Y1	M7Y0	MOB 7 Y-konumu
16	($10)	M7X8	M6X8	M5X8	M4X8	M3X8	M2X8	M1X8	M0X8	X-konumunun en soldaki bit
17	($11)	RCB	ECM	BMM	DEN	RSEL	Y2	Y1	Y0	Metne bakın
18	($12)	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Raster kaydı
19	($13)	LPX8	LPX7	LPX6	LPX5	LPX4	LPX3	LPX2	LPX1	Light pen X
20	($14)	LPY7	LPY6	LPY5	LPY4	LPY3	LPY2	LPY1	LPY0	Light pen Y
21	($15)	M7E	M6E	M5E	M4E	M3E	M2E	M1E	M0E	MOB etkin
22	($16)	—	—	RES	MCM	CSEL	X2	X1	X0	Metne bakın
23	($17)	M7YE	M6YE	M5YE	M4YE	M3YE	M2YE	M1YE	M0YE	MOB Y-yönünde genişleme
24	($18)	VM13	VM12	VM11	VM10	CB13	CB12	CB11	—	Bellek göstergeçleri
25	($19)	IRQ	—	—	—	ILP	IMMC	IMBC	IRST	Kesinti kaydı
26	($1A)	—	—	—	—	ELP	EMMC	EMBC	ERST	Kesinti etkinleştirme
27	($1B)	M7DP	M6DP	M5DP	M4DP	M3DP	M2DP	M1DP	M0DP	MOB-VERİ önceliği
28	($1C)	M7MC	M6MC	M5MC	M4MC	M3MC	M2MC	M1MC	M0MC	MOB çok-renkli seçimi
29	($10)	M7XE	M6XE	M5XE	M4XE	M3XE	M2XE	M1XE	M0XE	MOB X-yönünde genişleme
30	($1E)	M7M	M6M	M5M	M4M	M3M	M2M	M1M	M0M	MOB-MOB çarpışması
31	($1F)	M7D	M6D	M5D	M4D	M3D	M2D	M1D	M0D	MOB-DATA çarpışması
32	($20)	—	—	—	—	EC3	EC2	EC1	EC0	Dış renk
33	($21)	—	—	—	—	B0C3	B0C2	B0C1	B0C0	Zemin rengi 0
34	($22)	—	—	—	—	B1C3	B1C2	B1C1	B1C0	Zemin rengi 1
35	($23)	—	—	—	—	B2C3	B2C2	B2C1	B2C0	Zemin rengi 2
36	($24)	—	—	—	—	B3C3	B3C2	B3C1	B3C0	Zemin rengi 3
37	($25)	—	—	—	—	MM03	MM02	MM01	MM00	MOB çok-renkli 0
38	($26)	—	—	—	—	MM13	MM12	MM11	MM10	MOB çok-renkli 1
39	($27)	—	—	—	—	M0C3	M0C2	M0C1	M0C0	MOB 0 rengi
40	($28)	—	—	—	—	M1C3	M1C2	M1C1	M1C0	MOB 1 rengi
41	($29)	—	—	—	—	M2C3	M2C2	M2C1	M2C0	MOB 2 rengi
42	($2A)	—	—	—	—	M3C3	M3C2	M3C1	M3C0	MOB 3 rengi
43	($2B)	—	—	—	—	M4C3	M4C2	M4C1	M4C0	MOB 4 rengi
44	($2C)	—	—	—	—	M5C3	M5C2	M5C1	M5C0	MOB 5 rengi
45	($2D)	—	—	—	—	M6C3	M6C2	M6C1	M6C0	MOB 6 rengi
46	($2E)	—	—	—	—	M7C3	M7C2	M7C1	M7C0	MOB 7 rengi

NOT: Çizgiler hiçbir bağlantı olmadığını gösterir. Bunların hepsi "1" olarak okunur .

RENK KODLARI

D4	D3	D1	D0	ONALTILI	ONLU	COLOR
0	0	0	0	0	0	SİYAH
0	0	0	1	1	1	BEYAZ
0	0	1	0	2	2	KIRMIZI
0	0	1	1	3	3	TURKUAZ
0	1	0	0	4	4	MOR
0	1	0	1	5	5	YEŞİL
0	1	1	0	6	6	MAVİ
0	1	1	1	7	7	SARI
1	0	0	0	8	8	TURUNCU
1	0	0	1	9	9	KAHVERENGİ
1	0	1	0	A	10	AÇIK KIRMIZI
1	0	1	1	B	11	KOYU GRİ
1	1	0	0	C	12	ORTA GRİ
1	1	0	1	D	13	AÇIK YEŞİL
1	1	1	0	E	14	AÇIK MAVİ
1	1	1	1	F	15	AÇIK GRİ

6567 ZAMANLAMA SINIRLARI

ÖZEL	ÖZEL MİN.	TİP	ÖZEL MAK.
Saati kapat yüksek	465	484	500
Saati kapat düşük	475	494	510
RAS'a kadar saat düşük	150	171	190
RAS'a kadar saat yüksek	20	35	50
RAS düşük ila CAS düşük	25	46	65
CAS'a doğru saat yüksek	15	25	35
Saat AEC yüksek/düşük	15	33	50
CAS'tan gelen veriler		184	220
Ph0'dan gelen veriler	80	113	135
RAS kurulumuna eklenti	25	14	
RAS tutma işlemine eklenti	0	-15	
Eklenti/RAS kurulumu	35	48	
Eklenti/RAS tutma	30	36	45
Ph0'dan ekleme	85	97	
Ekleme/CAS↑ tutma	20	37	50
Ph0’dan BA	100	230	300
Kurulumdaki veriler/Ph0	60	42	
Tutmadaki veriler/Ph0	45	24	
Renk verisi kurulumu	45	30	
Renk verisi tutma	0	-17	
Ph girişi + nabız	50	43	
Ph girişi - nabız	65	58	
Vil		1.23	0.80
Vih	2.20	1.91	
Vol		0.52	0.55
Voh	2.40	3.03	

EK O

6581 SES ARABİRİM CİHAZI (SID)'NİN ÖZELLİKLERİ

KONU:

6581 Ses Arabirim Cihazı (SID). 65XX ve benzeri mikroişlemci ailesine uygun 3 sesli elektronik müzik synthesizer/ses efektleri üreticisi olan tek çiptir. SID kullanıcıya, yüksek frekans hassasiyeti kontrolu, perde (frekans) kontrolü ton rengi (armonik içeriği) ve dinamik (ses düzeyi) sağlar. Özel olarak planlanmış devresi ile donanımlarda az yer kaplar, bu yüzden ucuz müzik enstrümanları ile video oyunlarında kullanılır.

ÖZELLİKLERİ:

• 3 TONLU OSİLATÖRLER
  Frekans aralığı: 0-4 kHz

• HER OSİLATÖR İÇİN 4 DALGA BİÇİMİ
  Üçgen, dişli, değişken darbe, gürültü

• 3 GENLİK MODÜLATÖRÜ
  Genlik aralığı: 48 dB

• 3 SES ÜRETECİ
  Üslü karşılık
  Yükselme hızı : 2ms - 8s
  Düşme hızı : 6ms - 24s
  Durma seviyesi : 0 - en yüksek volüm
  Kaybolma hızı : 6ms - 24s

• OSİLATÖR SENKRONİZASYONU

• HALKA MODÜLASYONU

• PROGRAMLANABİLİR FİLTRE
  Kesme frekansı aralığı: 30Hz·12kHz
  12 dB/oktav Rolloff
  Alçak geçişli. Band geçişli
  Yüksek geçişli. Notch çıkışları
  Değişken Rezonans

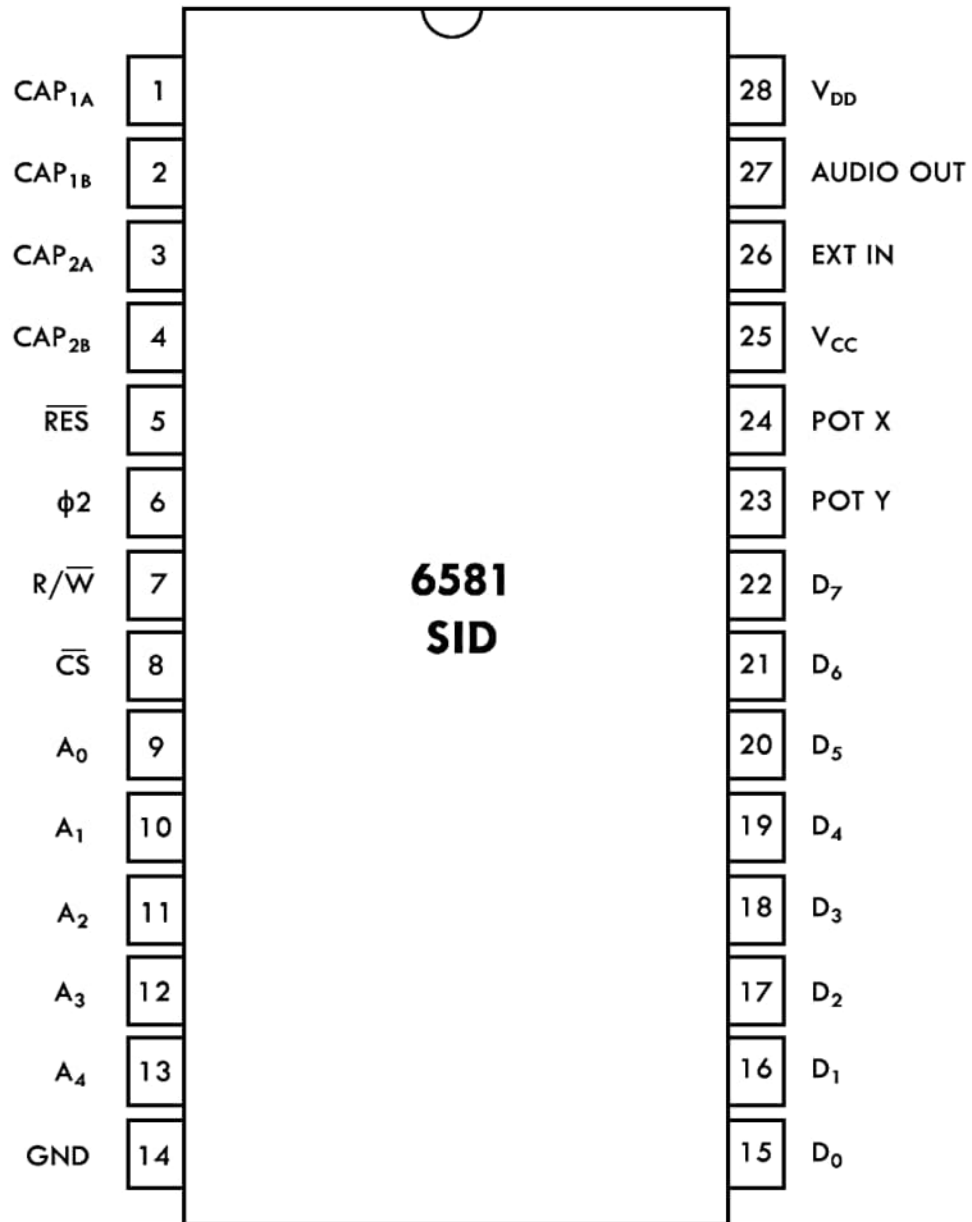
• ANA VOLÜM KONTROLU

• 2 A/D POT ARABİRİMİ

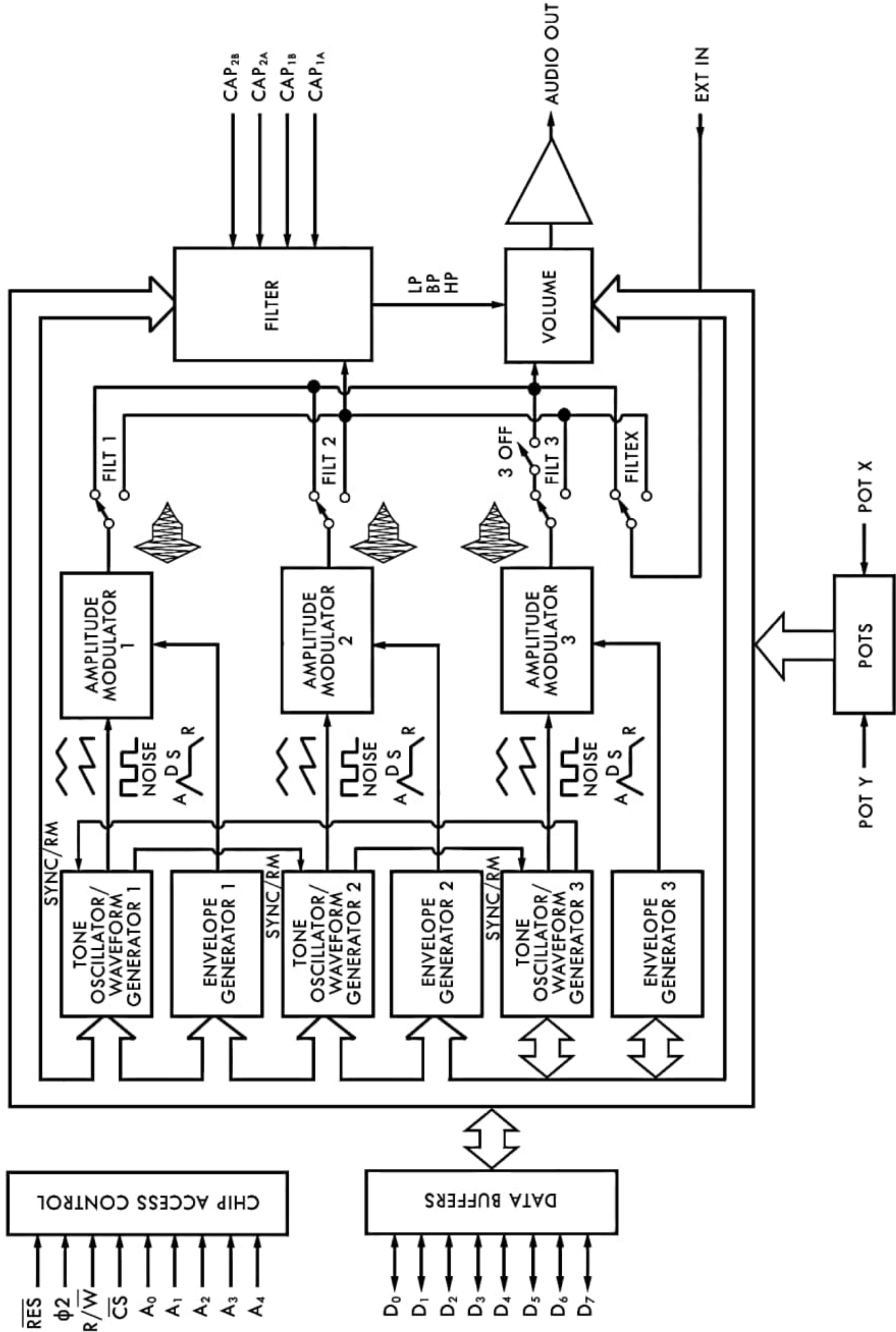
• RASTGELE SAYI/MODÜLASYON ÜRETECİ

• DIŞ SES GİRİŞİ

BACAK DİZİLİMİ



6581 BAĞLANTI ŞEMASI



TANIMLAMA

6581 'in kompleks sesler yaratabilmek için, bağımsız veya birleşik (dış ses kaynakları ile) kullanılan üç synthesizer “sesi” vardır. Her ses, bir ton osilatörü/dalga biçimi üreteci, bir ses üreteci ve bir genlik modülatörüne sahiptir. Ton osilatörü ses perdesinin, geniş bir aralıkta kontrol edilmesini sağlar. Osilatör seçilen frekansta dört değişik dalga biçimi üretir, her dalga biçiminin tek armonik içeriği vardır ve bu ton renginin basit olarak kontrol edilmesi sağlanır. Osilatör volüm dinamiği, ses üretecinin yönetiminde Genlik Modülatörü tarafından sağlanır. Ses üreteci tetiklendiğinde, programlanabilen alçalma ve yükselme volümünde bir genlik sesi yaratır. Bu üç sese ek olarak; kompleks ve dinamik ton renkleri üreten programlanabilir bir filtre vardır.

SID, mikroişlemcinin, üçüncü Osilatör ve üçüncü ses üretecinin değişen çıktılarını okuyabilmesine olanak sağlar. Bu çıktılar vibrato yaratımında frekans/filtre tarama ve benzeri efektlerin yaratılması amacıyla modülasyon bilgisi için kaynak olarak kullanılır. Üçüncü osilatör, oyunlar için rastgele sayı üreteci olarak da kullanılabilir. Her A/D (örneksel/sayısal) çeviricisi, SID’in potansiyometre ile bağlantısında arabirim görevi görür. Bu çeviriciler oyunlar için paddle veya müzik synthesizer'ında ön panel kontrolü olarak işlem görebilir. SID dıştan gelen ses sinyallerini işleyebilir. Böylelikle birden fazla SID çipi arka arkaya kullanılabilir ve çok sesli kompleks sistemler yaratılabilir.

SID KONTROL KAYITLARI

SIS içinde ses üretimini kontrol eden, 29 tane 8 bitlik kayıt vardır. Bu kayıtlar aşağıda verildiği gibi salt OKUNUR (READ-ONLY) veya salt YAZILIR (WRITE-ONLY) olarak kullanılırlar.

TABLO 1. SID KAYIT HARİTASI

	ADDRESS					REG#	DATA								REG NAME	REG
	A_4	A_3	A_2	A_1	A_0	(HEX)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Voice 1	TYPE
0	0	0	0	0	0	00	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE-ONLY
1	0	0	0	0	1	01	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE-ONLY
2	0	0	0	1	0	02	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE-ONLY
3	0	0	0	1	1	03	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE-ONLY
4	0	0	1	0	0	04	NOISE	⊓⊓	/|/|	/\/\	TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE-ONLY
5	0	0	1	0	1	05	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE-ONLY
6	0	0	1	1	0	06	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE-ONLY
															Voice 2	
7	0	0	1	1	1	07	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE-ONLY
8	0	1	0	0	0	08	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE-ONLY
9	0	1	0	0	1	09	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE-ONLY
10	0	1	0	1	0	0A	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE-ONLY
11	0	1	0	1	1	0B	NOISE	⊓⊓	/|/|	/\/\	TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE-ONLY
12	0	1	1	0	0	0C	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE-ONLY
13	0	1	1	0	1	0D	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE-ONLY
															Voice 3	
14	0	1	1	1	0	0E	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE-ONLY
15	0	1	1	1	1	0F	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE-ONLY
16	1	0	0	0	0	10	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE-ONLY
17	1	0	0	0	1	11	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE-ONLY
18	1	0	0	1	0	12	NOISE	⊓⊓	/|/|	/\/\	TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE-ONLY
19	1	0	0	1	1	13	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE-ONLY
20	1	0	1	0	0	14	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE-ONLY
															Filter	
21	1	0	1	0	1	15	—	—	—	—	—	FC_2	FC_1	FC_0	FC LO	WRITE-ONLY
22	1	0	1	1	0	16	FC_{10}	FC_9	FC_8	FC_7	FC_6	FC_5	FC_4	FC_3	FC HI	WRITE-ONLY
23	1	0	1	1	1	17	RES_3	RES_2	RES_1	RES_0	FILTEX	FILT 3	FILT 2	FILT 1	RES/FILT	WRITE-ONLY
24	1	1	0	0	0	18	3 OFF	HP	BP	LP	VOL_3	VOL_2	VOL_1	VOL_0	MODE/VOL	WRITE-ONLY
															Misc.	
25	1	1	0	0	1	19	PX_7	PX_6	PX_5	PX_4	PX_3	PX_2	PX_1	PX_0	POT X	READ-ONLY
26	1	1	0	1	0	1A	PY_7	PY_6	PY_5	PY_4	PY_3	PY_2	PY_1	PY_0	POT Y	READ-ONLY
27	1	1	0	1	1	1B	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0	OSC_3/RANDOM	READ-ONLY
28	1	1	1	0	0	1C	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	ENV_3	READ-ONLY

SID KAYIT TANIMI

SES 1

FREQ LO/FREQ HI (Alçak Frekans/Yüksek Frekans (00,01'inci Kayıtlar))

Bu kayıtlar, birlikte Osilatör 1'in frekansını doğrusal olarak kontrol eden 16 bitlik bir sayı oluşturur. Frekans şu denklemle elde edilir:

$F_{çıkış} = (F_n \times F_{CIK}/16777216)$ Hz

Burada F_n, Frekans kayıtlarının 16-bitlik sayı, saat ise •2 girişine (uç 6) uygulanan sistem saatidir. Standart 1,0 MHz saati için, frekans şöyle bulunabilir:

$F_{çıkış} = (F_n \times 0.059604645)$ Hz

SID'in frekans çözünürlüğü, herhangi bir tonlama ölçeği için yeterlidir.

PW ALÇAK/PW YÜKSEK (02,03 Kayıtları)

Bu kayıtlar, birlikte 12 bitlik bir sayı oluştururlar (PW YÜKSEK'İN 4-7 bitleri kullanılmaz). Osilatör 1'in darbe dalga biçiminin, darbe genişliğini doğrusal olarak kontrol eder. Darbe genişliği şu formülle belirlenir:

$PW_{çıkış} = (PW_n/40.95)\%$

Burada PW_n Darbe Genişliği kayıtlarındaki 12-bitlik sayıdır.

Darbe genişliği hassasiyeti ile, genişlik bir atlama olmayacak şekilde yumuşak bir biçimde ayarlanabilir. Fakat Osilatör 1'in darbe dalga biçiminin. Darbe Genişliği kayıtlarının duyulabilir etkisi olacak şekilde seçilmesi gerekir. Darbe genişliği kayıtlarındaki veya 4095($FF) değerleri. 2048($80) değeri kare dalga oluştururken sabit bir DC çıktısı üretirler.

KONTROL KAYDI (KAYIT 04)

Bu kayıt, Osilatör 1'in değişik çalışma biçimlerini seçen 8 kontrol bitini içerir.

GATE (Bit 0): GATE biti, Ses 1 için Ses Üretecini kontrol eder. Bu bit "1" olduğunda, ses üreteci harekete geçer ve YÜKSELME/DÜŞME/DURMA döngüsü başlatılır. Bit, "0" yapıldığında, KAYBOLMA döngüsü başlar. Ses Üreteci, Osilatör 1'in genliğini kontrol ettiğinden, GATE bitinin Osilatör 1 çıkışının istenildiği şekilde duyulabilmesi için de ayarlanması gerekir. Ses Üretecinin daha detaylı açıklaması ekin sonunda yer alıyor.

SYNC (Bit 1): (Senkronizasyon) SYNC biti "1" yapıldığında, Osilatör 1 'in ana frekansı ile Osilatör 3"ün ana frekansı "Güçlü senkronizasyon" etkisi yaratılarak senkronize edilir.

Osilatör 1'in frekansını Osilatör 3'e göre değiştirmek, Osilatör 3'ün frekansındaki ses 1 den. geniş bir aralıkta kompleks harmonik yapılar oluşturur. Senkronizasyon oluşabilmesi için, Osilatör 3'ün frekansının tercihan Osilatör 1'in frekansından daha küçük olarak, sıfır dışında bir frekans olarak seçilmesi gerekir. Ses 3'ün başka hiçbir değişkeninin senkronizasyon üzerinde etkisi yoktur.

RİNG MOD (Bit 2): (Halka Modülasyonu) Halka modülasyonu biti "1" yapıldığında, Osilatör 1'in üçgen dalga biçimi çıktısı, Osilatör 1'in frekansının, Osilatör 3'e bağlı olarak değiştirilmesi, ya da gong sesleri ve özel efektler için geniş aralıkta harmonik olmayan overtone yapılar oluştururlar. Halka modülasyonu dalga biçiminin duyulabilmesi için, Osilatör 1'in üçgen dalga biçimi ve Osilatör 3'ün frekansının da sıfırdan farklı olarak seçilmesi gerekir. Ses 3'ün başka hiçbir değişkeninin halka modülasyonuna etkisi yoktur.

TEST (Bit 3): TEST biti “1” yapıldığında, tekrar “0” yapılana kadar Osilatör 1'i susturur. Osilatör 1'in darbe dalga biçimi çıktısı bir DC düzeyinde tutulurken, görüntü dalga biçimi çıkışı da sıfırlanır. Normalde bu bit, test etme gereksinimleri için kullanılır, fakat, gerçek-zamanlı yazılım kontrolleri altında oldukça kompleks dalga biçimlerinin üretilmesine de izin vererek, Osilatör 1'in dış ortama göre senkronize edilmesi için kullanılır.

(Bit 4): Bit, “1” yapıldığında Osilatör 1'in üçgen dalga biçimi seçilir. Bu dalga biçiminin harmanisi oldukça düşüktür ve sesi flüt gibidir.

(Bit 5): Bit, “1” yapıldığında Osilatör 1'in dişli dalga biçimi seçilir. Dişli dalga biçimi tek ve çift harmonikler bakımından zengindir.

(Bit 6): Bit, “1” yapıldığında Osilatör 1'in darbe dalga biçimi seçilir. Bu dalga biçiminin harmonik içeriği, ton kalitelerinin değişimini sağlayan Darbe Genişliği kayıtları ile ayarlanır. Darbe genişliğinin gerçek zamanda değişimi, sese hareketlilik katan dinamik "faz" efekti verir. Değişik darbe genişliklerine yapılan ani sıçramalar ilginç harmonik diziler oluşturur.

GÜRÜLTÜ (Bit 7): Bit “1” yapıldığında. Osilatör 1'in Gürültü dalga biçimi seçilir. Çıktı, Osilatör 1'in frekansıyla değişen rasgele bir sinyaldir. Ses kalitesi, Osilatör 1 Frekans kayıtları yardımıyla alçak gürültüden beyaz gürültüye kadar değişir. Patlama, silah sesi, jet motoru sesi, rüzgâr ve diğer kompleks seslerin oluşumunda gürültü dalga biçimi kullanılır.

Osilatör 1’in duyulabilmesi için, çıkış dalga biçimlerinden birinin seçilmesi gerekir, buna karşılık Ses 1 çıkışının sessiz hale gelmesi için tekrar dalga biçimi seçimi yapılmasına gerek yoktur. Ses 1'in son çıktısındaki genliği, sadece ses üretecinin fonksiyonudur.

> **NOT:** Osilatör çıkış dalga biçimleri birbirine eklenemez. Aynı anda birden fazla dalga biçimi çıkışı seçilecek ise sonuç, dalga biçimlerinin mantıksal VE’lenmesi ile oluşacaktır. Bu teknik, yukarıda belirtilen dört biçime ek olarak, yeni dalga biçimleri üretebilir, fakat kullanılırken çok dikkatli olunması gerekir Gürültü dalga biçimi varken bir diğer dalga biçiminin seçilmesi Gürültünün susmasına sebep olur. Bunun sonucu olarak Gürültü çıkışı, RES (Uç 5). alt seviyeye getirilene kadar ya da TEST biti ile sıfırlanana kadar sessiz kalacaktır.

YÜKSELME/ALÇALMA (KAYIT 05)

Bu kaydın 4-7’nci bitleri (ATK0-ATK3), Ses üretecinin 1’den 16’ya kadar olan YÜKSELME hızlarını seçer, yükselme hızı, Ses 1'in, Ses üreteci harekete geçirildiğinde sıfırdan, en yüksek genliğe ulaşma hızını verir. 16 YÜKSELME hızı, Tablo 2'de verilmiştir.

0-3'üncü bitleri (DCY0-DCY3), Ses Üreteci için 1’den 16’ya kadar olan ALÇALMA hızlarını seçer. ALÇALMA döngüsü, YÜKSELME döngüsünden sonradır ve çıkışın en büyük genlikten DURMA seviyesine gelme hızını, ALÇALMA hızı belirler. 16 ALÇALMA hızı Tablo. 2'de verilmiştir.

DURMA/KAYBOLMA (Kayıt 06)

Bu kaydın 4-7'nci bitleri (STN0-STN3), Ses Üreteci için 1'den 16'ya kadar olan DURMA seviyelerini seçer. DURMA Döngüsü, ALÇALMA döngüsünü takip eder ve Ses 1'in çıkışı, Gate biti "1" olduğu sürece seçilen DURMA seviyesinde kalır. DURMA seviyeleri, 0 DURMA değeri sıfır genliği, 15($F) DURMA değeri en büyük genliği belirleyecek şekilde doğrusal olarak 16 basamaktır. DURMA değeri olarak seçilen 8 sayısı, SES 1'in, YÜKSELME döngüsünde erişilen en yüksek genliğin yarısına erişilinceye kadar DURMA'sını sağlar.

0-3 bitleri (RLS0-RLS3), Ses Üreteci için 1'den 16'ya kadar olan KAYBOLMA hızlarını seçer. KAYBOLMA döngüsü, Gate biti sıfırlandığında DURMA döngüsünden sonradır. Bu süre boyunca, Ses 1 çıkış seçilen KAYBOLMA hızında DURMA genliğinden, sıfır genişliğine düşer. 16 KAYBOLMA hızı, DÜŞME ile aynıdır.

NOT: Ses üreteci döngüsü, herhangi bir zamanda Gate biti ile değiştirilebilir. Örneğin, Yükselme döngüsü bitmeden Gate biti sıfırlanırsa, ulaşılan genlik seviyesinden itibaren KAYBOLMA döngüsü başlar. Ses tekrar tetiklendiğinde (KAYBOLMA döngüsü 0 seviyesine erişmeden) başka bir YÜKSELME döngüsü başlayacaktır. Bu teknik gerçek-zamanlı yazılım kontrolleri yardımıyla, kompleks genlik biçimlerinin üretiminde kullanılır.

Tablo 2. Değişim Hızları

DEĞER		YÜKSELME HIZI	DÜŞME/KAYBOLMA HIZI
ONLU	ONALTILI	(Süre/Döngü)	(Süre/Döngü)
0	0	2 ms	6 ms
1	1	8 ms	24 ms
2	2	16 ms	48 ms
3	3	24 ms	72 ms
4	4	38 ms	114 ms
5	5	56 ms	168 ms
6	6	68 ms	204 ms
7	7	80 ms	240 ms
8	8	100 ms	300 ms
9	9	250 ms	750 ms
10	A	500 ms	1.5 s
11	B	800 ms	2.4 s
12	C	1 s	3 s
13	D	3 s	9 s
14	E	5 s	15 s
15	F	8 s	24 s

NOT: Değişim hızları, 1.0MHz •2 saatine göre oluşturulur. Diğer •2 frekansları için, verilen hızı 1MHz •2 ile çarpın. Verilen hızlar, her döngüdeki süre miktarına karşılık gelir. Örneğin, 2 olarak verilen YÜKSELME hızı, YÜKSELME döngüsünün sıfırdan en yüksek genliğe, 16 ms'de ulaşacağını belirtir. DÜŞME/KAYBOLMA hızları, en büyük genlikten sıfıra düşme sürelerini gösterir.

SES 2

07-$0D kayıtları Ses 2'yi kontrol eder. Çalışmaları bakımından, aşağıda verilen özellikler dışında 00-06 kayıtlarına benzerler.

1) Seçildiğinde, SYNC (senkronizasyon biti) Osilatör 2 ile Osilatör 1'in senkronizasyonunu sağlar.

2) Seçildiğinde, RING MOD (halka modülasyon biti) Osilatör 2'nin üçgen dalga biçimini, Osilatör 2 ve 1'in birleşmesiyle oluşan halka ile değiştirir.

SES 3

$0E-$14 kayıtları Ses 3'ü kontrol eder. Çalışmaları bakımından, aşağıda verilen özellikler dışında 00-06 kayıtlarına benzerler.

1) Seçildiğinde, SYNC Osilatör 3'ün Osilatör 2 ile senkronizasyonunu sağlar.

2) Seçildiğinde. RING MOD'u, Osilatör 3'ün üçgen çıkışını osilatör 2 ve 3'ün birleşimiyle oluşan halka ile değiştirir.

Bir sesin tipik çalışma metodu, istenilen değişkenlerin seçimini içerir: frekans, dalga biçimi, efektler (SYNC. RING MOD), değişim hızları ve istenildiğinde sesin tetiklenmesi. Ses, herhangi bir zaman için belirli bir seviyede kalabilir ve Gate biti sıfırlanarak çıkışına son verilebilir. Her ses bağımsız değişkenleri ve tetiklemesiyle ayrı ayrı veya güçlü tek bir ses için beraber kullanılabilir.

FİLTRE

FC ALÇAK/FC YÜKSEK ($15, $16 Kayıtları)

Bu kayıtlar birlikte 11 bitlik bir sayı oluştururlar ve bu sayı programlanabilir filtrenin kesim frekansını kontrol eder. Kesim frekansları yaklaşık 30 Hz'den 12 KHz'e kadardır.

RES/FILT (Kayıt $17)

Bu kaydın 4-7'nci bitleri (RES0-RES3) filtrenin rezonansını kontrol eder. Rezonans, filtrenin kesim frekansındaki bileşenlerini yükseltip, daha kesin bir sesin oluşmasını sağlar. 0 rezonanstan, maksimum rezonansa (15 ya da $F) kadar düzgün aralıklarla, 16 rezonans şekli vardır. 0-3 bitleri filtrenin içinden geçecek sinyalleri belirler.

FILT 1 (Bit 0): "0" yapıldığında, çıkışta Ses 1 gözükür, filtrenin hiçbir etkisi yoktur. "1" yapıldığında ise, Ses 1 filtrelenir ve seçilen filtre değişkenlerine göre harmonikleri değiştirilir.

FILT 2 (Bit 1): Ses 2 için, bit 0'la aynı işi görür.

FILT 3 (Bit 2): Ses 3 için, bit 0'la aynı işi görür.

FILTEX (Bit 3): Dışarıdan ses girişi için, bit 0'la aynı işi yapar (26'ncı uç).

MOD/VOL (Kayıt $18)

Bu kaydın 4-7'nci bitleri değişik Filtre modları ve çıkış tiplerini seçerler.

LP (Bit 4): "1" yapıldığında, alçak frekans geçişli filtre seçilir. Verilen bir filtre giriş sinyali için, kesim frekansının birleşenleri değişmeden geçerken, bu frekanstan sonrakiler 12 dB/oktav oranında güç kaybına uğrarlar. Alçak frekans geçişli mod. tamamen gür-tok sesler üretilmesini sağlar.

BP (Bit 5): Bit 4 için anlatılanlar Bant geçişi olarak geçerlidir. Kesim frekanslarının alt ve üstündeki frekanslar, 6 dB/oktav oranında güç kaybederler. Bant geçişli modu, ince, açık ses üretir.

HP (Bit 6): Bit 4 için anlatılanlar, Yüksek Frekans geçişi olarak geçerlidir. Kesim frekansının üstündeki frekans bileşenleri değişmeden geçerken, altındaki frekans bileşenleri 12 dB/oktav oranında güç kaybederler. Yüksek frekans geçişli mod, teneke gibi sesler üretir.

3 OFF (Bit 7): “1” yapıldığında, Ses 3 çıkışı doğrudan işitilme yolundan ayrılır, böylece Ses 3 istenmeyen çıkışlar olmadan modülasyon amacıyla kullanılabilir.

> **NOT:** Filtre çıkış modları, beraber kullanılabilirler ve birden fazla olabilirler. Örneğin, LP ve HP modlarının beraber kullanımı ile Notch Filtre modu elde edilebilir. Filtrenin ses duyumunda herhangi bir etkisi olması için, Filtre çıkışlarından en az birisinin seçilmesi ve bir sesin filtre girişine verilmesi gerekir. Filtre sentezleme yoluyla kompleks ton renkleri üretimine izin verdiğinden SID’in en önemli elemanıdır (harmonik bakımından zengin bir giriş sinyalinin, belirli bazı frekanslarının yok edilmesinde Filtre kullanılır). Kesim frekansının gerçek zamanda değişimiyle, istenilen en iyi sonuçlar elde edilebilir.

BİT 0·3: (VOL0-VOL3) bitleri, son ses çıkışı için 1’den 16’ya kadar olan Volüm seviyeleri seçer. Çıkış volüm seviyeleri, hiç çıkış olmadığını gösteren seviye (0) ile maksimum çıkış arasında 16 doğrusal basmaktır. Bu kontrol, çok-çipli sistemlerde seviye dengelemesi için statik volüm kontrolü olarak kullanılabileceği gibi, Tremolo gibi dinamik Volüm efekti yaratımında da kullanılır. SID’ın ses üretebilmesi için, sıfırdan farklı bir Volüm seviyesinin seçilmesi gerekir.

TÜRLERİ

POTX (Kayıt $19)

Bu kayıt, POTX (uç 24)’e bağlı potansiyometrenin mikroişlemci tarafından okunmasını sağlar. Potansiyometre, 0’dan 255 ($FF)’e kadar değer alabilir. Direnç değerleri her zaman geçerli olup, her 512 •2 saat döngüsünde bir değişir. Pot ve kapasitör değerleri hakkında bilgi edinmek için Uç Tanımlama kısmına bakınız.

POTY (Kayıt $1A)

Potansiyometrenin POTY (Uç 23)'e bağlı olması dışında, POTX kaydı ile aynı işlevi görür.

OSC 3/RASGELE (Kayıt $1B)

Bu kayıt, mikroişlemcinin Osilatör 3'ün en büyük 8 çıkış bitini okumasını sağlar. Üretilen sayıların karakteristiği doğrudan seçilen dalga biçimine bağlıdır. Osilatör 3"ün dişli dalga biçimi seçildiyse, kayıt, hızı Osilatör 3 frekansıyla belirlenen. 0'dan 255’e kadar artan bir seri sayı gösterir. Üçgen dalga biçimi seçildiyse, çıkış 0’dan 255’e kadar artar. Sonra tekrar 0'a doğru azalır. Darbe dalga biçiminin seçiminde ise, çıkış 0 ile 255 arasında sıçrayacaktır. Gürültü dalga biçimi rasgele sayılar üretir. Bu yüzden kayıt, oyunlar için rasgele sayı üreteci olarak kullanılabilir. OSC 3 kaydının, zamanlama ve dizinleme için kullanıldığı çeşitli uygulamalar vardır, fakat en önemli işlevi modülasyon üreteci olarak kullanılmasıdır. Bu kayıt tarafından üretilen sayılar, yazılım ile, gerçek zamanda Osilatör veya Filtre frekans kayıtlarına veya darbe

genişliği kayıtlarına eklenebilir. Bu yolla birçok dinamik efekt üretebilir. Örneğin, siren benzeri sesler. OSC 3 üçgen çıkışının, bir diğer Osilatörün frekans kontrolüne eklenmesi ile elde edilebilir. Synthesizer "Örnekleme ve Tutma" efektleri, OSC 3 Gürültü çıkışının Filtre Frekans kontrol kayıtlarına eklenmesi ile üretilir. Vibrato, Osilatör 3 frekansının 7 Hz civarında tutulması ve OSC 3 üçgen çıkışının (belirli bir ölçü ile) başka bir osilatörün Frekans kontrolüne eklenmesi ile elde edilir. Osilatör 3'ün frekans değişimleri ve OSC 3 çıkışı ayarlamaları ile, sınırsız sayıda ses efekti yapılabilir. Normalde, Osilatör 3 modülasyon için kullanıldığında, Ses 3 işitsel çıkışının olmaması gerekir (3 OFF = 1).

ENV 3 (Kayıt $1C)

OSC 3 gibi çalışır; fakat bu kayıt, mikroişlemcinin Ses 3 zarf üreteci çıkışını okumasını sağlar. Bu çıkış, harmonik sesler, VAH-VAH ve benzeri efektlerin elde edilmesi için Filtre Frekansına eklenebilir. “Phaser” sesleri, çıktının osilatörün frekans kontrol kayıtlarına eklenmesi ile oluşur. Ses 3 üretecinin, bu kayıttan herhangi bir çıkış elde edebilmesi için tetiklenmesi gerekir. OSC 3 kaydı, daima osilatörün değişen çıktısını yansıtır ve ses üreteci tarafından hiçbir şekilde etkilenmez.

SID UÇLARININ AÇIKLAMALARI

CAP1A, CAP1B. (1, 2'nci Uçlar)/CAP2A, CAP2B, (3. 4'üncü Uçlar)

Bu uçlar, programlanabilen Filtre için gereksinilen iki entegre kapasitörün bağlantısı içindir. C1, 1 ve 2 uçları arasına, C2 ise 3 ve 4 uçları arasına bağlanır. İki kapasitörün değeri de aynıdır. Filtrenin, işitme aralığındaki (30 Hz-12 kHz) normal işlevi, C1 ve C2 için 2200 pF'lık değer ile sağlanır. Polystyrene kapasitörler tercih edildiği gibi, birden fazla SID çipinin olduğu polyphonic sistemlerde de birbirine uyumlu kapasitörler kullanılır.

Filtrenin frekans aralığı, kapasitör değerleri seçilerek özel uygulamalar için düzenlenebilir. Örneğin, ucuz bir oyunun yüksek frekans karşılığına ihtiyacı yoktur. Bu durumda, Filtrenin baş frekansları üzerinde daha fazla kontrol sağlamak amacıyla büyük C1 ve C2 değerleri kullanılabilir.

Filtrenin maksimum kesim frekansı;

$FC_{maks} = 2.6\ E\text{-}5/C$

ile belirlenir. Burada C kapasitör değeridir. Filtre aralığı maksimum kesim frekansından aşağı olmak üzere 9 oktavdır.

RES (Uç 5)

Bu TTL seviyeli giriş, SID'in reset kontrolüdür. En az on •2 döngüsü alçak seviyeye getirildiğinde, tüm iç kayıtlar sıfırlanır, işitsel çıkış sessiz hale geçer. Bu uç, normalde mikroişlemcinin sıfırlama hattına veya power-on-clear devresine bağlıdır.

*ϕ*2 (Uç 6)

Bu TTL seviyeli giriş, SID için ana saat görevini görür. Tüm osilatör frekansları ve değişim hızları bu saate göre ayarlanır. •2, SID ve mikroişlemci arasındaki veri transferini de kontrol eder. Veri, sadece •2 yüksek seviyede iken transfer edilebilir. Veri transferleri söz konusu olduğu surece •2, yüksek-aktif için uç normal olarak, 10 MHz'lık işlem frekansıyla sistem saatine bağlıdır.

R/W (Uç 7)

Bu TTL seviyeli giriş, SID ve mikroişlemci arasındaki veri transferinin yönünü kontrol eder. Çip seçimi durumuyla karşılaşıldığında, bu ucun yüksek seviyede olması, mikroişlemcinin istenilen SID kaydına veri yazmasına izin verir. Bu uç normal olarak, sistem Okuma/Yazma hattına bağlıdır.

CS (Uç 8)

Bu TTL seviyeli giriş, mikroişlemci ile SID arasındaki veri transferini kontrol eden, alçak seviyeli aktif çip seçicidir. Transfer olması için bu hattın alçak seviyede olması gerekir. Seçilen SID kaydından okuma yapılması için, CS'nin alçak, 2'nin yüksek ve R/W'ın da yüksek seviyede olması gerekir. Yazma işlemi için de CS'nin alçak, 2'nin yüksek, RW'in alçak seviyede olması zorunludur. Bu uç normalde, SID'nin sistem bellek haritasında görünmesine izin verecek şekilde adres çözümleme devresine bağlıdır.

A0-A4 (9-13 Uçları)

Bu TTL seviyeli girişler, 29 SID kaydından birini seçmek için kullanılır. 32 kaydından 1'ini seçebilmek için gerekli olan adresler sağlandığı halde, üç kayıt yerleşimi kullanılmaz. Bu üç yerleşimden birine yazmak istediğinizde, yerine getirilmez, okuma işlemi ise geçersiz veri elde eder. Bu uçlar normal olarak, mikroişlemcinin uygun adres hatlarına bağlanırlar. Böylece SID, tıpkı bellek gibi adreslenebilir.

GND (Uç 14)

En iyi sonuçlar için, SID ve güç kaynağı arasındaki toprak hattının diğer dijital devrelerle aradaki toprak hatlarından ayrı olması gerekir.

D0-D7 (15-22 Uçları)

Bu çift yönlü uçlar, SID ile mikroişlemci arasında, veri transferi için kullanılırlar. Giriş modunda TTL uyumlu, çıkış modunda ise 2 TTL yük sürebilecek kapasitededir. Veri tamponları, genellikle yüksek empedans off (çalışmama) durumundadır. Bir yazma işlemi süresince, veri tamponları off (giriş) durumunda kalırken, mikroişlemci bu hatlar üzerinden SID'e veri sağlar. Okuma işlemi süresince ise veri tamponları çalışır ve SID, bu hatlar üzerinden mikroişlemciye veri gönderir. Bu uçlar normal olarak, mikroişlemcinin uygun veri hatlarına bağlıdır.

POTX, POTY (23-24 Uçları)

Bu girişler, potansiyometrenin konumunu sayısallaştıran A/D (Ö/S) çeviricilerine bağlıdır. Çevirme işlemi POT'da toprağa bağlanan bir kapasitörün zaman sabitine bağlıdır. Bu kapasitör POT ucundan +5 volta bağlanan potansiyometre ile yüklenir. Eleman değerleri şu formülle bulunabilir:

RC = 4.7 E-4

Değerler 470 KΩ ve 1000 pF'dir. Her uç için ayrı bir pot ve kap gerektiğine dikkat ediniz.

V_{CC} (Uç 25)

SID V_{CC} ile güç kaynağı arasında gürültüyü azaltmak için toprak (GND) hattı gibi ayrı bir +5 VDC hattının olması gerekir. Uca yakın bir de kapasitör konmalıdır.

EXT IN (Uç 26)

Bu örneksel (analog) giriş, SID'in işitsel çıkışı ile dış işitsel sinyallerin karışmasını veya Filtrede işleme konulmasını sağlar. Ana kaynaklar ses, gitar veya org olabilir. Uçun giriş empedansı 100 KΩ civarındadır. Doğrudan uca uygulanan bir sinyalin 6 Voltluk DC seviyeyi sürebilmesi ve 3 voltluk maksimum minimum (peak to peek) aralığını aşmaması gerekir. DC seviye farklılığı yüzünden oluşabilecek herhangi bir karışıklığı önlemek amacıyla, dış sinyallerin 1-10 μF civarında bir elektrolitik kapasitörle, EXT IN ucuna AC için bağlanması gerekir. Direct audio path'ın kazancı bir olduğundan EXT IN, birden fazla SID çipinin zincirleme olarak çıkışlarının karışmasında kullanılabilir. Kullanılacak SID çiplerinin sayısı son çıkışta mümkün olabilecek gürültü ve bozulma miktarı ile belirlenir. Çıkış Volüm kontrolunun, üç SID sesinden başka dış girişleri de etkilediğine dikkat ediniz.

AUDIO ÇIKIŞI (Uç 27)

Bu açık-kaynak tamponu, SID'ın üç ses, Filtre ve dış girişten oluşan son işitsel çıkışıdır. Çıkış seviyesi çıkış Volüm kontrolü tarafından seçilir ve 6 Volt DC seviyesinde, 2 Volt maksimum minimum aralığı vardır. Normal bir çalışma için AUDIO OUT ucundan toprağa bir kaynak direncinin konması gerekir. Standart çıkış empedansı olarak 1 KΩ'luk bir direnç tercih edilir.

SID çıkışı 6 volt DC seviyesinde olduğundan, bir işitsel büyütücüye 1-10 μF civarında kapasitörle AC için bağlanması gerekir.

V_{DD} (Uç 28)

V_{CC}'de olduğu gibi, bir +12 VDC hattının, SID V_{DD}'ye baglanması ayrıca bir bypass kapasitörünün kullanılması gereklidir.

6581 SID KARAKTERİSTİKLERİ

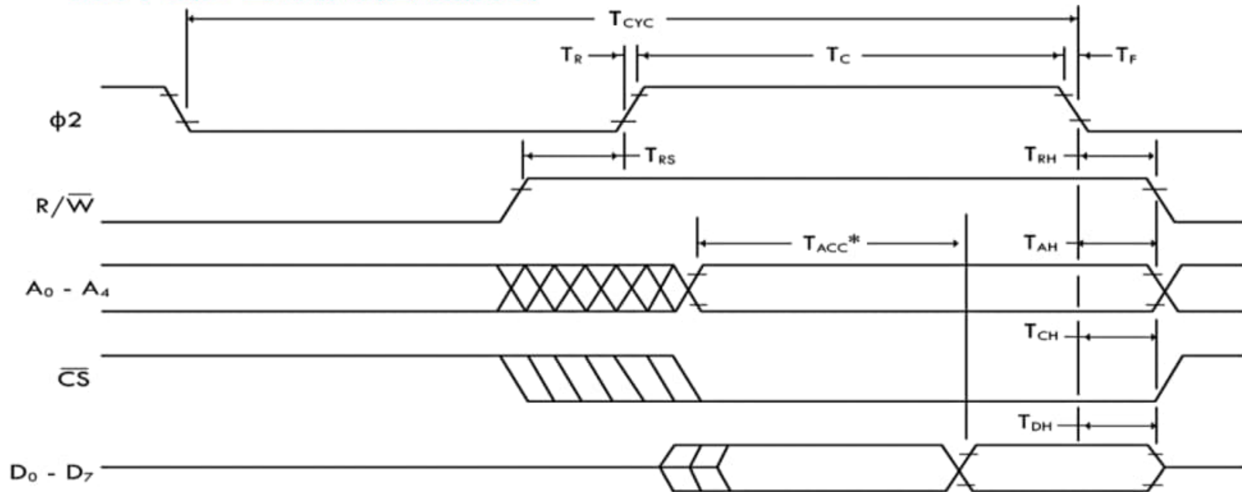
MUTLAK MAKSİMUM DEĞERLENDİRMELER

ÇALIŞMA DURUMU	SEMBOL	DEĞER	BİRİM
Güç kaynağı	V_{DD}	-0.3'den +17'ye	VDC
Güç kaynağı	V_{CC}	-0.3'den +7'ye	VDC
Giriş voltajı (örneksel)	V_{INA}	-0.3'den +17'ye	VDC
Giriş voltajı (sayısal)	V_{IND}	-0.3'den +7'ye	VDC
Çalışma ısısı	T_A	0'dan +70'e	°C
Depolama ısısı	T_{STG}	-55'den +150'ye	°C

ELEKTRİKSEL KARAKTERİSTİKLER
(V_{DD} = 12 VDC ±5%, V_{CC} = 5 VDC ±5%, T_A = 0'dan 70°C'ye)

KARAKTERİSTİK		SEMBOL	MİN.	TİP	MAKS.	BİRİM
Giriş Yüksek Voltaj Giriş Alçak Voltaj	(RES, 2, R/W, CS A0-A4, D0-D7)	V_{IH} V_{IL}	2 -0.3	— —	V_{CC} 0.8	VDC VDC
Giriş Kaçak Akım Üç Durum (Off)	(RES, 2, R/W, CS A0-A4; V_{IN} = 0 - 5 VDC) (D0-D7; V_{CC} = maks.)	I_{IN} I_{TSI}	— —	— —	2.5 10	µA µA
Giriş Kaçak Akım	V_{IN} = 0.4 - 2.4 VDC					
Çıkış Yüksek Voltaj	(D0-D7; V_{CC} = min, 1 load = 200 µA)	V_{OH}	2.4	—	V_{CC} - 0.7	VDC
Çıkış Alçak Voltaj	(D0-D7; V_{CC} = maks, 1 load = 3.2 mA)	V_{OL}	GND	—	0.4	VDC
Çıkış Yüksek Akım	(D0-D7; Kaynak, V_{OH} = 2.4 VDC)	I_{OH}	200	—	—	pA
Çıkış Alçak Akım	(D0-D7; Batan, V_{OL} = 0.4 VDC)	I_{OL}	3.2	—	—	mA
Giriş Kapasitansı	(RES, 02, R/W, CS, A0-A4, D0-D7)	C_{IN}	—	—	10	pF
Pot Tetikleme Voltajı	(POTX, POTY)	V_{POT}	—	$V_{CC}/2$	—	VDC
Pot Çekme Akımı	(POTX, POTY)	I_{POT}	500	—	—	µA
Giriş Empendansı	(EXT IN)	R_{IN}	100	150	—	kΩ
Ses Giriş Voltajı	(EXT IN)	V_{IN}	5.7 —	6 0 .5	6.3 3	VDC VAC
Ses Çıkış Voltajı	(AUDIO OUT; 1 kΩ load, volume = maks.) Bir ses açık: Tüm sesler açık:	V_{OUT}	5.7 0.4 1.0	6 0.5 1.5	6.3 0.6 2.0	VDC VAC VAC
Güç Kaynağı Akımı	(V_{DD})	I_{DD}	—	20	25	mA
Güç Kaynağı Akımı	(V_{CC})	I_{CC}	—	70	100	mA
Güç Dağılım	(Toplam)	P_D	—	600	1000	mW

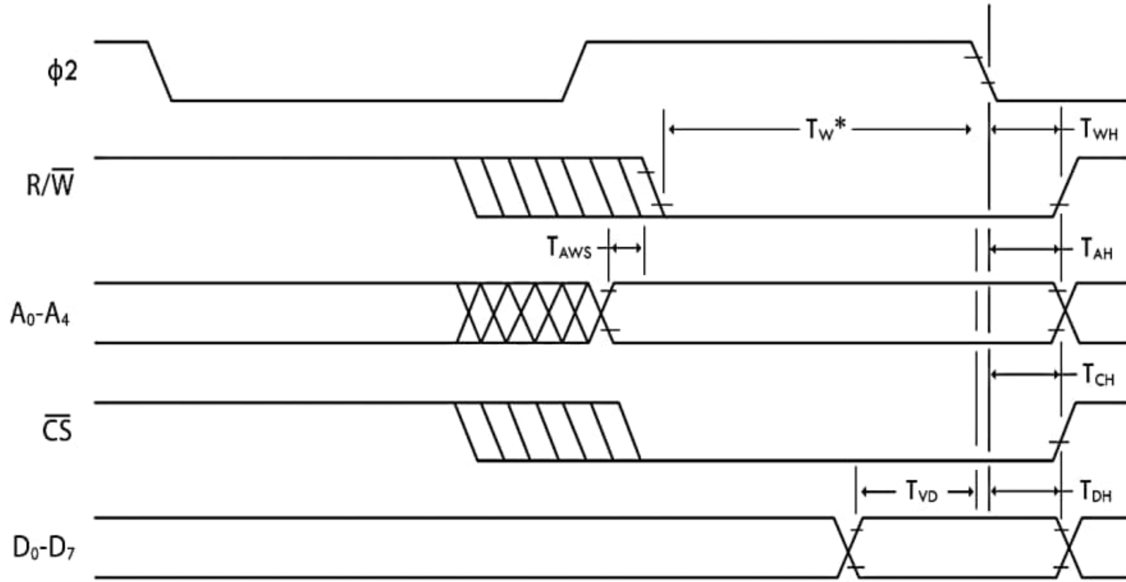
6581 SID ZAMANLAMASI



T_{ACC}, 2, $\overline{CS}$, A_0-A_4 sinyallerinin en son oluşumundan ölçülür.

OKUMA DÖNGÜSÜ

SEMBOL	İSİM	MİN.	TİP	MAKS.	BİRİM
T_{CYC}	Saat Döngü Süresi	1	—	20	µs
T_C	Saat Yüksek Vurum genişliği	450	500	10,000	ns
T_R, T_F	Saat Yükselme/Düşme suresi	—	—	25	ns
T_{RS}	Okuma Kurulum süresi	0	—	—	ns
T_{RH}	Okuma Tutma Süresi	0	—	—	ns
T_{ACC}	Erişim Süresi	—	—	300	ns
T_{AH}	Adres Tutma Süresi	10	—	—	ns
T_{CH}	Çip Seçimi Tutma Süresi	0	—	—	ns
T_{DH}	Veri Tutma Süresi	20	—	—	ns



T_W, $\phi 2$, $\overline{CS}$, $R/\overline{W}$ sinyallerinin son oluşumundan ölçülür.

YAZMA DÖNGÜSÜ

SEMBOL	İSİM	MİN.	TİP	MAKS.	BİRİM
T_W	Yazma Vurum Genişliği	300	—	—	ns
T_{WH}	Yazma Tutma Süresi	0	—	—	ns
T_{AWS}	Adres Kurulum Süresi	0	—	—	ns
T_{AH}	Adres Tutma Süresi	10	—	—	ns
T_{CH}	Çip Seçimi Tutma Süresi	0	—	—	ns
T_{VD}	Geçeni Veri	80	—	—	ns
T_{DH}	Veri Tutma Süresi	10	—	—	ns

EŞİT GÜÇLÜ MÜZİK ÖLÇEK DEĞERLERİ

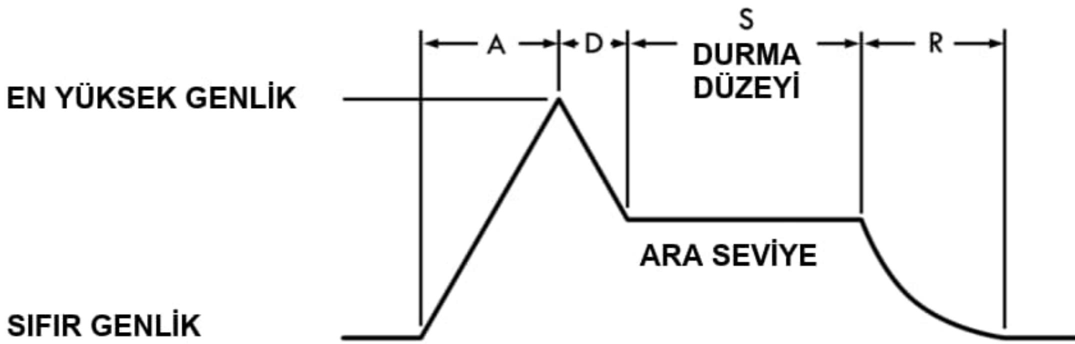
Ek E'deki tabloda, nota üretimi için SID Osilatör frekans kontrol kayıtlarında saklanması gereken sayısal değerlerin listesi bulunur. Notalar C, D, E, F, G, A, B ve C, D#, F#, G#, A#, 12 yarı-ton (nota) ile bir oktavdan oluşmuştur. Her yarı tonun frekansı, 2'nin 12'nci kökü ($\sqrt[12]{2}$) çarpı ile evvelki tonun frekansı şeklinde bulunur. Tablo, 1.02 MHz'lik $\phi 2$ saatine göre oluşturulmuştur. Diğer ana saat frekansları için Kayıt Tanımlama kısmındaki eşitliğe bakınız. Seçilen ölçü, A-4 = 440 Hz olan konser perdesidir. Bu ölçeğin transpozisyonu ve equal-tempered ölçeğinden farklı ölçeklerde mümkündür.

Ek E'deki tablo eşit güçlü bir ölçü üretiminde basit ve hızlı bir metot olduğu halde, 192 bayta ihtiyacı olduğundan bellekte fazla yer kaplar.

Nota değeri algoritmik olarak belirtilerek bellek yeri kazanılabilir. Her noktanın, bir sonraki oktavdaki frekans değerinin yarısı olacağı gerçeğini göz önüne alarak nota bakma tablosu. 96 girişten 12 girişe düşebilir. Çünkü her oktavda sadece 12 nota vardır. Eğer 12 giriş (24 bayt) 8 oktavın 16 bitlik değerlerini içeriyorsa (C-7'den B-7'ye), alçak oktavlardaki notalar 8'inci oktavdan uygun notanın seçimi ve 16 bitlik sayının her oktav farkı için 2 ile bölünmesi ile bulunabilir. İki ile bölünme, değerin sağa kayması demek olduğundan hesaplama çok basit bir yazılım programı ile yapılabilir. B-7, Osilatörlerin kapasiteleri dışında olduğu halde, bu değer hesaplamalar için tabloda bulunur (B-7'nin en soldaki biti, bu bitin kayma olmadan evvel ELDE bitinde elde edilmesi gibi özel yazılımlar gereksinecektir). Her notanın 8 oktavın hangisinden ve 12 yarı-tonun hangisi olduğunu belirtir bir biçimde olması gerekir. 12 yarı-tondan birinin seçimi için 4,8 oktavdan birisinde olduğunu belirtmek için de 3 bit gerektiğinden bilgi bir bayt içine sığar. Burada alçak basamak nybble'ı yarı-ton seçiminde (bakma tablosunu adresleyerek) büyük basamak baytı da tablo değerinin kaç defa sağa kaydırılması gerektiğini belirten bölme programı tarafından kullanılır.

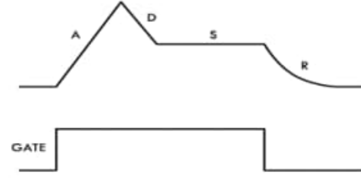
SID ZARF (ENVELOPE) ÜRETEÇLERİ

Dört bölümlü YDDK (ADRS) (Yükselme, Düşme, Durma, Kaybolma) ses üreteci, elektronik müzikte, genlik kontrol kolaylığı ve esneklik arasında, en iyi konumun bulunması amacıyla geliştirilmiştir. Ses değişkenlerinin uygun seçimi ile birçok enstrümanın taklidi yapılabilir. Durma seviyeli enstrümanlara en güzel örnek kemandır. Kemanda ses yavaş yavaş en yüksek seviyeye çıkar, buradan da ara bir seviyeye düşer. Kemancı bu düzeyi, istendiği sürece devam ettirir, daha sonra volümün yavaşça yok olması sağlanır. Bu sesin görüntüsü aşağıda verilmiştir.



Volüm değişiklikleri aşağıda verilen, tipik değişim hızları YDDK (ADRS) ile kolayca tekrar üretilebilir.

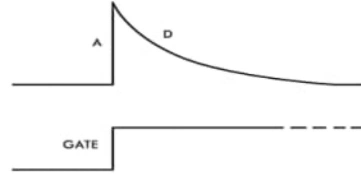
YÜKSELME:	10 ($A)	500 ms
DÜŞME:	8	300 ms
DURMA:	10 ($A)	
KAYBOLMA:	9	750 ms



Tonun ara DURMA düzeyinde istenildiği kadar uzun bir süre tutulabileceğine dikkat ediniz. GATE biti sıfırlanmadan ton, susmaya başlamayacaktır.

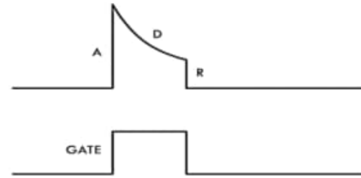
Davul, cembals ve gonglar gibi percussion enstrümanları ile klavyeli enstrüman olan piyano ve harpsichord için daha değişik bir ses biçimi verilmiştir. Bu seste çok ani bir yükselme ve gene ani bir düşme durumu vardır. Percussion enstrümanlarında sabit bir kalma seviyesi yoktur. Aşağıda tipik bir cymbal sesi gösterilmiştir.

YÜKSELME:	0	2 ms
DÜŞME:	9	750 ms
DURMA:	0	
KAYBOLMA:	9	750 ms



Tonun, en yüksek seviyeye erişildikten sonra GATE bitinin sıfırlanmasına bakılmadan düşmeye başlamasına dikkat ediniz. Piyano ve harpsıchordların ses genlikleri daha komplekstir. Fakat YDDK ile kolaylıkla üretilebilir. Bu enstrümanlar en yüksek seviyeye, tuşa vurulduğu an erişirler. Tuş basılı kaldığı sürece, genlik yavaş yavaş azalır, ses tamamen susmadan parmak tuştan kaldırılınca genlik aniden O olur. Bu ses aşağıda verilmiştir.

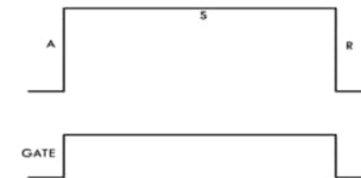
YÜKSELME:	0	2 ms
DÜŞME:	9	750 ms
DURMA:		
KAYBOLMA:		6 ms



Tonun GATE biti, sıfırlanıncaya kadar düştüğünü, bu andan sonra aniden sıfır olduğuna dikkat ediniz.

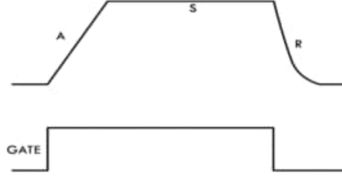
En basit ses org sesidir. Bir tuşa basıldığında, ton aniden tam volüme çıkar ve orada kalır. Tuş bırakıldığında ton aniden sıfır volümüne düşer. Bu ses de aşasıda verilmiştir.

YÜKSELME:	0	2 ms
DÜŞME:	0	6 ms
DURMA:	15 ($F)	
KAYBOLMA:	0	6 ms



SID'in esas gücü, akustik enstrüman taklitlerinin yanı sıra orijinal sesler üretebilmesinden gelir. YDDK hiçbir gerçek enstrümana karşılık gelmeyen sesler de üretebilir. Buna en güzel örnek “backwards” sesidir. Bu ses, yavaş yükselme ve hızlı düşme karakteristiği ile teybe yapılan bir kaydın geri çalınması ile elde edilebilecek sese çok benzer. Ses aşağıda verilmiştir.

YÜKSELME:	10 ($A)	500 ms
DÜŞME:	0	6 ms
DURMA:	15 ($F)	
KAYBOLMA:	3	72 ms

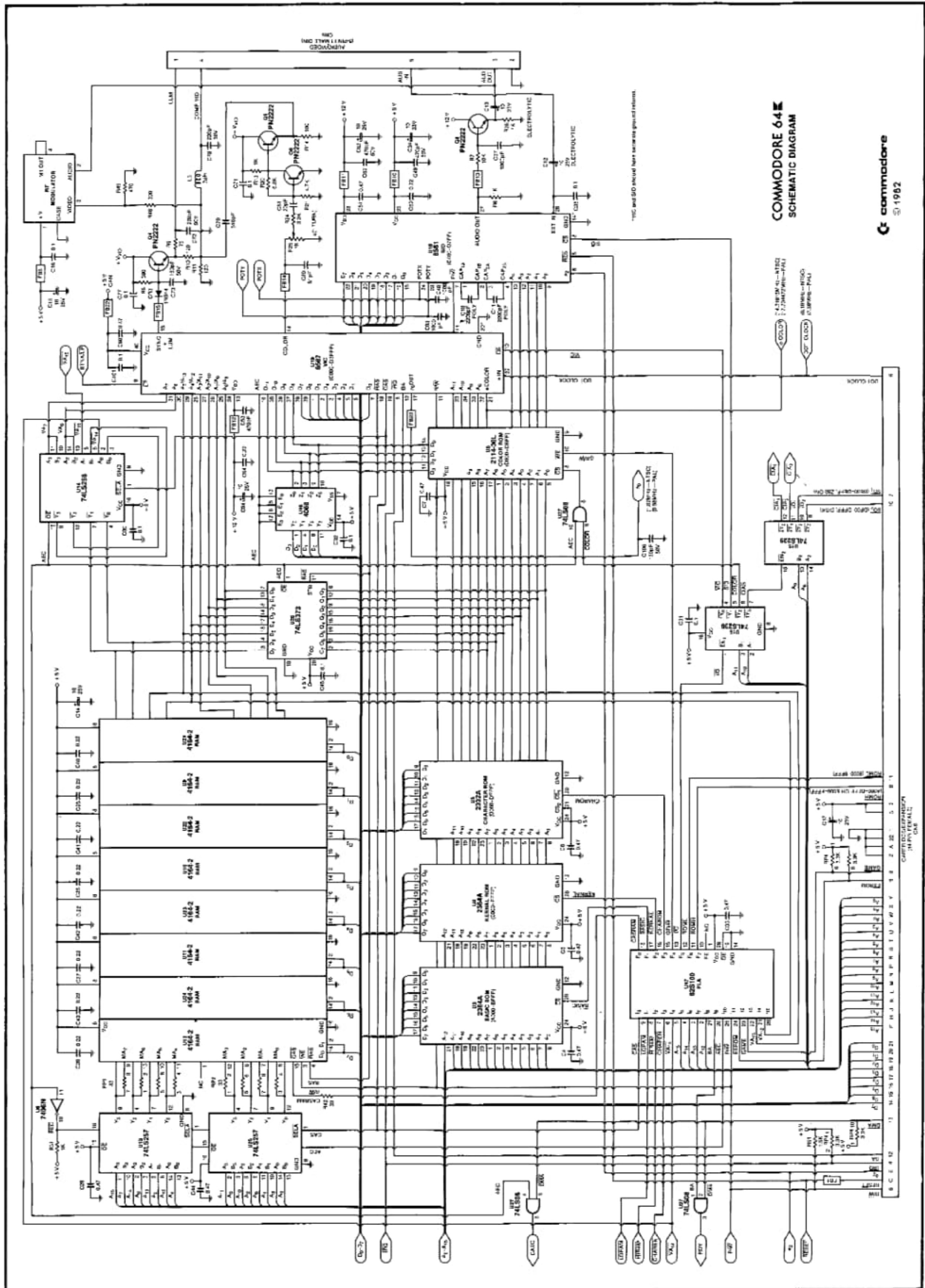


Bir enstrümanın genlik biçimini bir başkasının karmaşık yapısına uygulayarak değişik sesler üretilebilir. Bu teknikle bilinen akustik enstrümanların simülasyonu da yapılabilir. Genelde istenilen sesin üretilmesi için, değişik değişim oranları ve harmonik içeriklerinin denenmesi gerekir.

TİPİK 6581/SID UYGULAMASI

-12V
+5V
1.0 MHz
CLOCK CIRCUIT
RESET CIRCUIT
+5V
2200 pF POLYSTYRENE
2200 pF POLYSTYRENE
V_{DD}
V_{CC}
CAP_{1A}
CAP_{1B}
CAP_{2A}
CAP_{2B}
AUDIO OUT
EXT IN
ELECTROLYTIC
1.0 µF
1K
AUDIO OUT
ELECTROLYTIC
1.0 µF
AUDIO IN
V_{CC}
ϕ0 IN
RES
ϕ2 OUT
R/W
A_n
A_5
ADDRESS DECODING OR ADDRESS LINE
650X MPU
A_4 A_3 A_2 A_1 A_0
RES
ϕ2
R/W
CS
6581 SID
POT X
POT Y
470K
+5V
1000 pF
PADDLES
470K
+5V
1000 pF
A_4 A_3 A_2 A_1 A_0
GND
D_7 … D_0
D_7 … D_0
V_{SS}

EK P
COMMODORE 64 ELEKTRONİK DEVRE ŞEMASI



COMMODORE 64
SCHEMATIC DIAGRAM



EK R

MİNİ SÖZLÜK

ADSR (YDDK)	Attack/Decay/Sustain/Release envelopu. (Yükselme/Düşme/Durma/Kaybolma).
Attack (Yükselme)	Notanın en yüksek volüme ulaştığı hız.
Binary (İkili)	Taban-2 sayı sistemi
Boole Operatörleri	Mantıksal operatörler.
Byte (Bayt)	Bellek yeri. 8 bite bir bayt denir.
CIA	Complex Interface Adapter: Kompleks arabirim cihazı
DDR	Data Direction Register: Veri Yön Kaydı.
Decay (Düşme)	Notanın en yüksek volümden, durma volümüne düşme hızı.
Decimal (Onlu)	Taban-10 sayı sistemi.
E	Matematiksel sabit (yaklaşık 2.71828183'e eşittir).
Envelope	Bir notanın volümünün zamana bağlı şekli.
FIFO	First-In/First-Out: İlk giren, ilk çıkar.
Hexadecimal (Onaltılı)	Taban-16 sayı sistemi.
Integer	Tamsayı (kesir noktası yok).
Jiffy Saati	İç, aralık zamanlayıcısı.
NMI	Non-Maskable Interrupt: Maskelenemez kesinti.
Octal (Sekizli)	Taban-8 sayı sistemi.
Operand	Parametre.
OS (İS)	Operating System: İşletim sistemi.
Piksel	Ekran üzerindeki çözünürlük noktası.
Queue (kuyruk)	Tek-dosya hattı. Sıra bekleme durumu.
Register (Kayıt)	özel bellek saklanım yeri.
Release (Kaybolma)	Notanın durma volümünden sıfır volüme düşme hızı.
ROM	Read-Only Memory (Salt-Okunur Bellek).
SID	Ses Arabirim Aygıtı.
Sustain (Durma)	Notanın durması için volüm düzeyi.
Syntax (Yazım)	Programın cümle yapısı.
VIC-II	Video Arabirim Çipi.

COMMODORE 64 HIZLI REFERANS KARTI

BASİT DEĞİŞKENLER

Tür	*İsim Aralığı*
Gerçek	XY = 1.70141183E+38
	= 2.93873588E-39
Tamsayı	XY% = 32767
Dizin	XY$ 0 ile 255 karakter arasında

X bir harftir (A-Z), Y bir harf veya rakamdır (0-9). Değişken adları 2 karakterden fazla olabilir, ancak yalnızca ilk ikisi tanınır.

DİZİ DEĞİŞKENLERİ

Tür	*İsim*
Tek Boyut	XY(5)
İki Boyut	XY(5,5)
Üç Boyut	XY(5,5,5)

Gerektiğinde en fazla on bir elemandan oluşan diziler (0-10 arası alt dizinler) kullanılabilir. On birden fazla elemana sahip dizilerin BOYUTLANDIRILMASI gerekir.

CEBİRSEL İŞLEMLER

=	Değişkene değer atar
-	Olumsuzlaştırma
↑	Üs alma
*	Çarpma
/	Bölme
+	Toplama
-	Çıkarma

İLİŞKİSEL VE MANTIKSAL OPERATÖRLER

=	Eşit
<>	Eşit değil
<	Küçüktür
>	Büyüktür
<=	Küçük veya eşit
>=	Büyük veya eşit
NOT	Mantıksal "Değil"
AND	Mantıksal "Ve"
OR	Mantıksal "Veya"

İfade doğru ise 1'e, yanlış ise 0'a eşittir.

SİSTEM KOMUTLARI

LOAD "İSİM"	Teypten bir program yükler
SAVE "İSİM"	Teybe bir program saklar
LOAD "İSİM",8	Diskten bir program yükler
SAVE "İSİM",8	Diske bir program saklar
VERIFY "İSİM"	Programın hatasız kaydedildiğini doğrular
RUN	Bir programı çalıştırır
RUN XXX	Bir programı XXX satırından çalıştırır
STOP	Çalışmayı durdurur
END	Çalışmayı sonlandırır
CONT	Programın durdurulduğu satırdan program yürütmeye devam eder
PEEK(X)	Bellek konumu X'in içeriğini döndürür
POKE X,Y	X konumunun içeriğini Y değerine değiştirir
SYS xxxxx	Xxxx'ten başlayarak bir makine dili programını çalıştırmak için atlar
WAIT X,Y,Z	Program, Z ile EOR ve Y ile AND işlemi yapıldığında X konumunun içeriği sıfırdan farklı olana kadar bekler
USR(X)	X değerini makine dili alt yordamına iletir

DÜZENLEME VE BİÇİMLENDİRME KOMUTLARI

LIST	Tüm programı listeler
LIST A-B	A satırından B satırına kadar listeler
REM Message	Yorum mesajı listelenebilir ancak program yürütülürken göz ardı edilir
TAB(X)	PRINT ifadelerinde kullanılır. Ekrandaki X konumlarını boşluk bırakır
SPC(X)	X boşluklarını satıra yazdırır
POS(X)	Geçerli imleç konumunu döndürür
CLR/HOME	İmleci ekranın sol köşesine konumlar
SHIFT CLR/HOME	Ekranı temizler ve imleci "Ana Sayfa" konumuna getirir
SHIFT INST/DEL	Mevcut imleç konumuna boşluk ekler
INST/DEL	Geçerli imleç konumuna boşluk ekler
CTRL	Sayısal renk tuşuyla kullanıldığında metin rengini seçer. PRINT ifadesinde kullanılabilir.
CRSR Keys	İmleci ekranda yukarı, aşağı, sola, sağa hareket ettirir
Commodore Key	SHIFT tuşuyla kullanıldığında büyük /küçük harf ve grafik görüntüleme modu arasında seçim yapar. Sayısal renk tuşuyla kullanıldığında isteğe bağlı metin rengini seçer

DİZİLER VE DİZENLER

DIM A(X,Y,Z)	A için maksimum alt dizinleri ayarlar; A(0,0,0) noktasından başlayarak (X+l)*(Y+1)*(Z+1) öğeleri için alan ayırır
LEN(X$)	X$'deki karakter sayısını döndürür
STR$(X)	X'in sayısal değerini bir dizeye dönüştürür
VAL(X$)	X$'in sayısal olmayan ilk karakterine kadar olan sayısal değerini döndürür
CHR$(X)	Kodu X olan ASCII karakterini döndürür
ASC(X$)	X$'in ilk karakteri için ASCII kodunu döndürür
LEFT$(A$,X)	A$'ın en soldaki X karakterini döndürür
RIGHT$(A$,X)	A$'ın en sağdaki X karakterini döndürür
MID$(A$,X,Y)	X karakterinden başlayarak A$'ın Y karakterini döndürür

GİRİŞ/ÇIKIŞ KOMUTLARI

INPUT A$ or A	Ekrana "?" yazdırır ve kullanıcının bir dize veya değer girmesini bekler
INPUT "ABC",A	Mesajı yazdırır ve kullanıcının değer girmesini bekler. Ayrıca A$ girişi de yapılabilir.
GET A$ or A	Kullanıcının bir karakter değeri girmesini bekler, RETURN'e gerek yoktur
DATA A,"B",C	READ ifadesi tarafından kullanılabilecek bir değer kümesini başlatır
READ A$ or A	Sonraki DATA değerini A$ veya A'ya atar
RESTORE	Veri işaretçisini sıfırlayarak VERİ listesinin tekrar OKUNMASINI başlatır
PRINT "A= ";A	"A=" dizesini ve A değerini YAZDIRIR, ";" boşlukları bastırır, "," verileri bir sonraki alana sekmeler

PROGRAM AKIŞI

GOTO X	X satırına kadar dallar
IF A=3 THEN 6	Eğer iddia doğruysa, ifadenin aşağıdaki kısmını yürüt. Eğer yanlışsa, bir sonraki satır numarasını yürüt.
FOR A=1 TO 6 STEP 2: NEXT	FOR ile karşılık gelen NEXT arasındaki tüm ifadeleri, A'nın 1'den 6'ya 2'şer adım ilerlemesiyle yürütür. Aksi belirtilmediği sürece adım boyutu 1'dir
NEXT A	Döngünün sonunu tanımlar. A isteğe bağlı
GOSUB 20	20.satırdan başlayarak alt programa dallar
RETURN	Alt programın sonunu işaretler. En son GOSUB'u takip eden ifadeye döner
ON X GOTO A,B	Dallar listedeki X. satır numarasına gider. Eğer X = 1 ise dallar A'ya gider, vb.
ON X GOSUB A,B	Listedeki X. satır numarasındaki alt yordama dallar

PROGRAM ÖRNEKLERİ

Ekrana Desen Çizdirme

```
10 PRINT CHR$(205.5+RND(1)); : GOTO 10
```

Araba Yarışı

```
10 rem araba yarisi
20 rem vic=53248
30 for y=1 to 63*2
40 read dt
50 poke 831+y,dt
60 next
65 poke 646,12
75 print chr$(147);
100 rem  - yaratik 1
101 data 0,0,0,0,0
102 data 0,0,0,0,0,0,0,0
103 data 0,0,0,0,0,0,0,0
104 data 0,0,0,0,10,0,0,21
105 data 160,0,85,80,2,85,80,42
106 data 170,104,170,170,168,154,170,168
107 data 154,170,152,86,170,84,84,0
108 data 84,84,0,84,16,0,16,0,0,0,0
109 rem  - yaratik 2
110 data 0,0,0,0,0
111 data 0,0,0,0,0,0,0,0
112 data 0,0,0,0,0,0,0,0
113 data 0,0,0,0,160,0,10,84
114 data 0,5,85,0,5,85,128,41
115 data 170,168,42,170,170,42,170,166
116 data 38,170,166,21,170,149,21,0
117 data 21,21,0,21,4,0,4,0,0,0,0
200 for n=0 to 7
210 poke 2040+n,13:next
220 poke 53248+0,88
230 poke 53248+2,88
240 poke 53248+4,88
250 poke 53248+6,88
260 poke 53248+8,88
270 poke 53248+10,88
280 poke 53248+12,88
290 poke 53248+14,88
300 poke 53248+23,255
305 poke 53248+29,255
310 poke 53248+28,255
320 poke 53248+16,255
330 poke 53248+21,255
335 poke 53248+37,0
340 for n=0 to 7
```

```
345 poke 53248+39+n,n
348 if n=6 then poke 53248+39+n,8
349 if n=0 then poke 53248+39+n,9
350 poke 53248+n*2+1,32+25*n:next
400 for n=0 to 7
410 for x=255+88 to 24 step -10
420 z=peek(53248+16)
430 if x>255 then 450
440 poke 53248+16,z and not 2^n
450 if x<256 then 470
460 poke 53248+16,z or 2^n
470 if x>255 then poke 53248+2*n,x-255
480 if x<256 then poke 53248+2*n,x
490 next:poke 2040+n,14:next
500 for y=0 to 24:poke 1024+34+40*y,81:n
ext
505 poke 214,10:print:poke 211,19
508 print "bahsinizi koyun"
510 poke 214,11:print:poke 211,19
520 print"klavyeye bas"
530 poke 214,12:print:poke 211,19
540 print "yarisi baslat"
600 get kb$:if kb$="" then 600
700 r=int(rnd(1)*8)
710 poke 53248+r*2,peek(53248+r*2) + 5
715 if peek(53248+r*2)>250 then goto 800
720 goto 700
800 poke 646,1
801 if peek(53248+0*2)>250 then print "k
ahverengi araba kazandi!"
802 if peek(53248+1*2)>250 then print "b
eyaz araba kazandi!"
803 if peek(53248+2*2)>250 then print "k
irmizi araba kazandi!"
804 if peek(53248+3*2)>250 then print "t
urkuaz araba kazandi!"
805 if peek(53248+4*2)>250 then print "p
embe araba kazandi!"
806 if peek(53248+5*2)>250 then print "y
esil araba kazandi!"
807 if peek(53248+6*2)>250 then print "a
cik kahverengi araba kazandi!"
808 if peek(53248+7*2)>250 then print "s
ari araba kazandi!"
809 get kb$:if kb$="" then 809
820 goto 65
```

Mirror Horror

```
4 rem dsiclaimer - dont remove border ga
me crash
10 rem "mirror horror #5-9"
20 poke 53281,0:poke 53280,0
30 poke 646,5:print chr$(147):gosub 200
40 poke 646,1:a$=""
42 for cnt=1 to
35:a$=a$+chr$(157):next:a$=a$+chr$(17):r
em left 35
100 poke 214,4:print:poke 211,3
110 print "use mirror to kill monsters!
      ";a$;
112 print "
";a$;
115 print "use cursors to move over mirr
orrs  ";a$;
120 print "
      ";a$;
125 print "space-bar to tilt mirrors
      ";a$;
130 print "
      ";a$;
135 print "dont let monsters touch you
      ";a$;
136 print "
      ";a$;
138 print "controls:
      ";a$;
139 print "
      ";a$;
140 print "[cursors][z][x][spacebar][ret
urn]  "
144 get keyb$
149 if keyb$="" then goto 144
154 keyb$=""
197 for cnt=1 to 100:print:next
198 poke 646,1:print chr$(147):gosub
200:goto 300
199 :rem
200 for cnt=0 to 19:rem "game frame"
201 poke 1024+19-cnt+40*24,160
202 poke 1024+19-cnt+40*0,160
203 poke 1024+20+cnt+40*24,160
204 poke 1024+20+cnt+40*0,160
206 poke
1024+00+40*(12+int(12/19*cnt)),160
207 poke 1024+00+40*(12-
int(12/19*cnt)),160
208 poke
1024+39+40*(12+int(12/19*cnt)),160
209 poke 1024+39+40*(12-
int(12/19*cnt)),160
210 next:return
300 :rem
301 dim m(4):rem "mirror angle"
302 m(1)=78:m(2)=67:m(3)=77:m(4)=66
```

```
303 g(1)=107:g(2)=113:g(3)=115:g(4)=114
400 :rem
401 for cnt=1 to 10:rem "set mirrors"
402 mx=int(rnd(0)*38)+1
403 my=int(rnd(0)*23)+1
404 m=int(rnd(0)*4)+1
405 pk=peek(1024+mx+40*my)
406 if pk<> 32 then 402
407 poke 1024+mx+40*my,m(m):next
410 :rem
411 for cnt=1 to 10:rem "set enemy"
412 ex=int(rnd(0)*38)+1
413 ey=int(rnd(0)*23)+1
414 pk=peek(1024+ex+40*ey)
415 if pk<>32 then 412
416 poke 1024+ex+40*ey,88
417 poke55296+ex+40*ey,2:next
420 :rem
421 rem "set gun"
422 gx=int(rnd(0)*38)+1
423 gy=int(rnd(0)*23)+1
424 g=int(rnd(0)*4)+1
425 pk=peek(1024+gx+40*gy)
426 if pk<>32 then 422
427 poke 1024+gx+40*gy,g(g)
428 poke55296+gx+40*gy,6
500 :rem
501 rem "set player"
502 px=int(rnd(0)*38)+1
503 py=int(rnd(0)*23)+1
504 pk=peek(1024+px+40*py)
505 if pk<>32 then 502
600 :rem
601 pk=peek(1024+px+40*py)
602 poke1024+px+40*py,81:rem rst stp
603 poke1024+px+40*py,pk:dx=0:dy=0
604 get keyb$:if keyb$="" then 602
605 if keyb$=" " then gosub 700
606 if keyb$=chr$(29) then dx=+1
607 if keyb$=chr$(157)then dx=-1
608 if keyb$=chr$(17) then dy=+1
609 if keyb$=chr$(145)then dy=-1
610 if keyb$="x" or keyb$="z" then gosub
800
611 if keyb$=chr$(13) then gosub 900
617 s=peek(1024+px+dx+40*(py+dy))
618 if s=160 then 602
619 if s=88 then goto 2000
620 px=px+dx:py=py+dy:goto 601
699 :end:goto 800
700 :rem
702 rem "adjust"
704 ad=peek(1024+px+40*py)
706 if ad=m(1) then ad=m(2):goto 725
708 if ad=m(2) then ad=m(3):goto 725
710 if ad=m(3) then ad=m(4):goto 725
712 if ad=m(4) then ad=m(1):goto 725
714 if ad=g(1) then ad=g(2):goto 725
```

```
716 if ad=g(2) then ad=g(3):goto 725
718 if ad=g(3) then ad=g(4):goto 725
720 if ad=g(4) then ad=g(1):goto 725
725 poke 1024+px+40*py,ad:return
800 :rem
801 if keyb$<>"z" then 810:rem pick mirr
802 for cnt=1 to 4
804 pk=peek(1024+px+40*py)
806 if pk<>m(cnt) then 809
808 poke1024+px+40*py,32:mr=mr+1
809 next
810 if keyb$<>"x" then 825:rem place mir
812 if mr=0 then 825
814 pk=peek(1024+px+40*py)
816 if pk<>32 then 825
818 m=int(rnd(0)*4)+1:mr=mr-1
820 poke 1024+px+40*py,m(m)
825 return
900 :rem
910 pk=peek(1024+gx+40*gy)
911 :rem
912 if pk=107 then bx=+1:by=0
914 if pk=115 then bx=-1:by=0
916 if pk=114 then bx=0:by=+1
918 if pk=113 then bx=0:by=-1
920 x=gx:y=gy
921 pk=peek(1024+x+40*y)
922 if pk=32 then poke 1024+x+40*y,102
924 if pk=88 then sc=sc+1000
925 if pk=88 then poke 1024+x+40*y,86
926 x=x+bx:y=y+by
928 pk=peek(1024+x+40*y)
930 if pk<>78 then 950
934 if bx=+1 and by=0 then d=4:
936 if bx=-1 and by=0 then d=3:
938 if bx=0 and by=+1 then d=2:
941 if bx=0 and by=-1 then d=1:
950 if pk<>77 then 960
952 if bx=+1 and by=0 then d=3
954 if bx=-1 and by=0 then d=4
956 if bx=0 and by=+1 then d=1
958 if bx=0 and by=-1 then d=2
960 if d=1 then bx=+1:by=0
962 if d=2 then bx=-1:by=0
964 if d=3 then bx=0:by=+1
966 if d=4 then bx=0:by=-1
999 if pk<>160 then 921
1000 print chr$(147):
1010 x=2:y=2:x$=" good work":gosub 3000
1020 x=3:y=4:x$="you killed":gosub 3000
1040 x$=str$(sc/1000)+" enemies of 10 po
ssible"
1050 x=4:y=6:gosub 3000:x=5:y=8
1060 x$="more plays?":gosub 3000
1070 get keyb$:if keyb$="" then 1070
1080 if keyb$="y" then run
1090 if keyb$="n" then end
1099 goto 1070
```

```
2000 :rem
2001 poke 214,12:print:poke 211,13
2002 print chr$(18);"           ";chr$(1
46)
2004 poke 214,13:print:poke 211,13
2006 print chr$(18);" game over ";chr$(1
46)
2008 poke 214,14:print:poke 211,13
2009 print chr$(18);"           ";chr$(1
46)
2010 for wt=0 to 400:next wt:
2011 print chr$(147);
2012 print " lets try again "
2013 for wt=0 to 400:next wt:run
3000 poke 214,y:print:poke 211,x
3010 print x$;:return
```

NOTLAR :

COMMODORE 64 KULLANICI KILAVUZU HAKKINDA. . .

Olağanüstü renkler... ses sentezi... grafikler... hesaplama yetenekleri... en son teknolojilerin sinerjik birleşimi. Bu özellikler, Commodore 64'ü sınıfının en gelişmiş kişisel bilgisayarı yapar.

Commodore 64 Kullanıcı Kılavuzu, daha önce hiç bilgisayar kullanmamış olsanız bile bilgisayar dünyasına başlamanıza yardımcı olur. Açık ve adım adım talimatlarla, BASIC dili ve Commodore 64'ün sayısız farklı alanda nasıl kullanılabileceği hakkında bilgi edinirsiniz.

Mikrobilgisayarlara aşina olanlar için, gelişmiş programlama bölümleri ve ekler, Commodore 64'ün gelişmiş özelliklerini ve bu genişletilmiş özelliklerden en iyi şekilde nasıl yararlanılacağını açıklar.

COMMODORE 64 KULLANICININ REHBERİ

İLK BASKI 1982

SON BASKI 2025

DÜZENLEME Recep MUM

REVİZYON R.22.11.2025.0008